AFRL-HE-WP-TR-2006-0123

A REVIEW AND REAPPRAISAL OF ADAPTIVE HUMAN-COMPUTER INTERFACES IN COMPLEX CONTROL SYSTEMS

Waldemar Karwowski
Center for Industrial Ergonomics
University of Louisville
Louisville, Kentucky 40292-2001

Michael Haas
Human Effectiveness Directorate
Warfighter Interface Division
Wright-Patterson AFB, Ohio 45433-7022

Gavriel Salvendy
School of Industrial Engineering
Purdue University
West Lafayette, Indiana 47907-2023

August 2006

Interim Report from the period May 2003 to January 2004

Air Force Research Laboratory
Human Effectiveness Directorate
Warfighter Interface Division
Collaborative Interfaces Branch
Wright Patterson AFB OH 45433

NOTICE

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2006-0123

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

FOR THE DIRECTOR

//signed//

DANIEL G GODDARD
Chief, Warfighter Interface Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
1 Aug 2006	Interim	May 2003 – January 2006

4. TITLE AND SUBTITLE
A review and reappraisal of adaptive human-computer interfaces in complex control systems

5a. CONTRACT NUMBER
In-House

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62202F

6. AUTHOR(S)
Waldemar Karwowski*
Michael Haas**
Gavriel Salvendy***

5d. PROJECT NUMBER
7184

5e. TASK NUMBER
08

5f. WORK UNIT NUMBER
72

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Louisville*
Louisville KY 40292
Purdue University***
West Lafayette IN 47907

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Materiel Command**
Air Force Research Laboratory
Human Effectiveness Directorate
Warfighter Interface Division
Collaborative Interfaces Branch
Wright Patterson AFB OH 45433-7022

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/HECP

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-HE-WP-TR-2006-0123

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
AFRL/PA Cleared 12-30-03, AFRL/WS 03-3297.

14. ABSTRACT
This report reviews literature through 2003 on the design of adaptive human-computer interfaces for the control of complex systems and their application in a variety of domains, including control of technological systems, process control, aviation systems, flight navigation, database design and management, and computer software development and utilization. It is concluded that a significant portion of the current application literature focuses on the user-model construction, the control mechanisms, and technical aspects of the interface architecture. The cognitive aspects of the user-model that are utilized to drive system adaptation are in most cases intuitive and underdeveloped. Also, human information perception and cognitive processing is seldom considered in the design of adaptive human-computer interfaces. Application of soft computing methodologies and techniques is one of the more promising new approaches in this area of research.

15. SUBJECT TERMS
Adaptive Interfaces, crew systems, HCI, HMI, control, adaptive functional allocation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Michael W. Haas
U	U	U	SAR	108	19b. TELEPONE NUMBER (*Include area code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI-Std Z39-18

This page intentionally left blank.

Table of Contents

LIST OF FIGURES

1. INTRODUCTION

This report reviews the recent literature on the design of adaptive human-computer interfaces for control of complex systems and their application in a variety of domains, including control of technological systems, process control, aviation systems, flight navigation, database design and management, and computer software development and utilization. According to Rothrock (2002), an ***adaptive interface*** autonomously adapts its displays and available actions to current goals and abilities of the user by monitoring user status, the system task, and the current situation. In other words, an ***adaptive user interface*** aims to adapt itself to the characteristics of individual users and their specific ways of performing tasks while using an application system (Kühme, 1993; Houlier, Grau and Valot, 2003). It is widely accepted that such an adaptation requires the interface to maintain embedded models of users and tasks. It should also be noted that the adaptive interface acts primarily as an intelligent intermediary that dynamically allocates the tasks and task components to either system or operator (Morris, Rouse and Ward, 1988; Chignell and Hancock, 1988; Frey, Rouse and Garris, 1992).

1.1 Adaptive systems, automation, control, and interfaces

Despite the long history of research on adaptive control and considerable practical success of adaptive strategies, a satisfactory definition of *adaptation* remains elusive (Rouse, 1988, 1990; Zames, 1998; Hettinger and Haas, 2003). According to Wickens (1992), *adaptive systems* are those in which some characteristic of the system changes or adapts, usually in response to measured or inferred characteristics of the human user. *Adaptive systems* are systems, which can alter aspects of their structure, functionality or interface in order to accommodate the differing needs of individuals or groups of users and the changing needs of users over time (Benyon 1987; Andes and Rouse., 1992). A common idea is that adaptation occurs when parameters inside a controller vary in response to changes in the environment. According to Zames (1998), there is no clear separation between the concepts of adaptation and nonlinear feedback, or between research on adaptive control and nonlinear stability.

The other two important ideas in the context of this review are the concepts of *adaptive automation* and *adaptive interfaces*. Hilburn, Parasuraman, and Mouloua (1995) define adaptive automation as *the real-time allocation of functions between human operator and automated system*. According to Parasuraman (2002), the *adaptive automation* involves the human-computer systems in which the “division of labor” and/or the interface between human and machine agents is not fixed at system design, but can vary dynamically during system operations. An *adaptive interface* is one where the appearance, function or content of the interface can be

changed by the interface (or the underlying application) itself in response to the user's interaction with it (Keeble and Macredie, 2000). Rouse, Geddes, and Curry (1988) defined an *adaptive interface* from a goal-oriented perspective. The reason for its existence is for the operator to remain in control and be provided with aiding that adapts to current needs and capabilities, in order to utilize human and computer resources optimally and, thereby, enhance overall performance.

Other definitions of *adaptive interfaces* differ due to their intended primary application. For example, according to Hettinger (2003), an adaptive interface consists of an ensemble of displays and controls whose features can be made to change in real time in response to variations in parameters indexing the state of the user—either some internal state, such as level of cognitive workload or engagement in a particular task
(e.g. Pope et al., 1995), and/ or a relevant external task-related condition, such as the nature, number and priority of tasks to be performed within a given unit of time (e.g.
Mulgund et al., 2002). According to Arai et al. (1993), an *interactive adaptation interface* is the interface that is changing according the given task considering the user features such as skill level, techniques, characteristics, physical condition, etc.

An *adaptive support system* facilitates the human decision-making judgments by adapting support to the high-level cognitive needs of the users, task characteristics, and decision contexts (Fazlollahi et al., 1997). Langley (1998) stated that *adaptivity* is a software artifact that improves its ability to interact with a user by constructing a user model based on partial experience with that user. The term *active user interface* has also been used in the subject literature. According to Brown and Santos (1999), the *active user interfaces* serve as actuators in the human-machine interface, and allows the user to interact with the computer in a naturalistic/symbiotic manner. Furthermore, an *intelligent interface* was defined as *smoothly changing its behavior to fit with users' knowledge, abilities and preferences, usually with advanced dialogue (and multimodal), capabilities* (Hook, 1998). According to Takahashi et al. (1994) an adaptive interface is an intelligent interface that can accommodate the form of human machine interaction according to the mental and physical state of operator.

Finally, Soulard (1992) has introduced the concept of the *self-adaptive interfaces*, arguing that taking into account both physiological and cognitive human factors enables the system to propose dynamically a set of pertinent data according to the operational context and to the operator mental state. The goal is to facilitate and optimize his task especially in critical situations. The main difference between self-adaptive interfaces and adaptable interfaces is that the adaptable interfaces are defined during the design of the interface taking in consideration only predefined levels of competence. On the other hand, a system with self-adaptive interfaces adapts during run time the nature, the kind of communication devices and the logic of the interactions to the characteristics of the task and to the physiological and cognitive state of the

human operator.

Zames (1998) proposed re-examination of the notions of *adaptation* and *learning,* on both conceptual and design levels. The main ideas behind this approach are outlined as follows. Adaptation and learning involve the acquisition of information about the plant (i.e., object to be controlled). Better performance requires more information. The performance function determines the nature of the information. For feedback control the appropriate notions of information are metric, locating the plant in a metric space in one of a set of neighborhoods of possible plants. Metric information can be quantified. The measures of metric complexity most frequently used for this purpose are (1) metric dimension (inverse n-width), and (2) metric entropy. The object of identification is to get this metric information, which takes time to acquire. The minimum time needed to acquire it is related to the metric complexity of a priori data. There are two monotonicity principles:

- **Monotonicity Principle 1.**
  Information obtainable at any given time about behavior at some future target date is a monotone increasing function of time.
- **Monotonicity Principle 2.**
  Optimal performance is a monotone increasing function of relevant information.

The non-adaptive (robust) control performance is designed or optimized on the basis of a priori information. On the other hand, adaptive control is based on a posteriori information, and uses the extra information to achieve improved performance. To flesh out these ideas, a number of mathematical results will be outlined. Most of them have been obtained during the past ten years or so, and many require further development.

Recent control literature indicates that with the increase in computational capability, computational strategies of control are directed more toward intelligent behavior that is increasingly being employed as a tool within an adaptive control technique. Major control research focus is on fuzzy logic, neural networks, genetic algorithms, and rule-based learning. Often, in the development of a particular system, more than one of these tools can be employed in a hybrid fashion (Warwick, 1996). According to An et al. (1994), any *intelligent module* must be able to modify its behavior in response to its interaction with the current environment, and to be able to associate its current experiences with similar events that have happened in the past. This means that an intelligent module must be able to adapt and in a local manner. Within the context of intelligent control, and intelligent controller must be able to modify its strategy according to its current performance and this modification will affect the output of the controller for similar inputs (Tolle and Ersfi, 1992).

2. ADAPTIVE INTERFACES IN AVIATION

Early studies by Amalberti and his co-workers on the human-machine interfaces (Amalberti and Menu, 1985; Menu, Amalberti and Santucci, 1986; Menu and Amalberti, 1988) formed a basis for development of the adaptive interfaces in military aviation. Examples of such studies include cognitive modeling of the fighter aircraft process control and development of an intelligent on-board assistance systems (Amalberti and Deblon, 1992), decision-making under time-pressure in air combat missions (Amalberti, 1991), reasoning model of the fighter pilots (Amalberti, 1992), etc. Some of the more recent studies in this area are discussed below.

2.1 Dynamic adaptive interfaces in aircraft aviation

Bennett (2001) conducted a preliminary investigation of dynamic adaptive interfaces in the domain of aviation. The primary aim of this study was the examination of the potential performance decrements associated with an inconsistency and unpredictability of three adaptive interfaces. The *standard, candidate,* and *adaptive interfaces* were evaluated in their effectiveness in supporting Air Force pilots to complete a precision low-level navigation task. The *standard interface* includes: 1) controls (throttle and joystick) displays (a horizontal situation display (HSD), 2) an attitude directional indicator (ADI), and a 3) head-up display (HUD) in de-clutter mode). . The *candidate interface* contained an alternative control (a force-reflecting stick) and an alternative display (configural flight director (CFD) - HUD). The force-reflecting stick controls the pilot's input (i.e. amount of force required to implement the control input) as a function of the plane's deviation from the optimal flight path. As opposed to the standard interface, which presents current values for task-relevant variables, the computational aiding component of the CFD-HUD calculates commanded control inputs (roll, pitch, and throttle) necessary to maintain the aircraft's position on the optimal flight path. The representational aiding component of the CFD-HUD combines this information in a centralized and easily interpretable display format.

For the *adaptive interface*, the *standard HUD* was used under conditions of the optimal aircraft performance (deviations from the optimal flight path of less than 500-ft laterally or 50-ft vertically; and deviations between the ETA and tuning goal of less than 10 sec). The *candidate HUD* indicates that an aircraft is outside the above performance criteria. Two additional display sets were included to the adaptive interface. An ADI presented a vertical velocity and angle of attack indicators. Second one was an HSD similar to the HSD--moving map display in the F-15E. This display presented an overhead perspective of the waypoints, course, and aircraft's position relative to them.

The *configural display* (CFD) includes both a geometric format and a visual reference point: a rectangular box and a watermark symbol. The component of the rectangle serves as a reference to ground, whereas the dashed component serves as a reference to the sky. This

aspect of the display serves as a cue for the aircraft--ground relation. Deviations of the aircraft from the optimal flight path result in movements of the rectangle relative to the fixed reference point. A deviation in altitude is represented by a vertical displacement of the rectangle. A deviation in heading is represented by rotation in the rectangle. The CFD HUD used the airspeed calculations employed in the standard interface.

The *candidate interface* condition also contains a force-reflective haptic stick. The side-stick controller was connected to a McFadden hydraulic control loader, which allowed numerous aspects of stick feel to be modified in real time. The force-reflective stick was programmed to provide a command input of sorts. A pilot who initiated inappropriate control inputs (those that would move the aircraft away from the optimal flight path) would receive haptic feedback in the form of increased resistance. The analysis of different interface impact on the navigation task showed significant performance advantages in the quality of route navigation with the candidate and adaptive interfaces relative to the standard interface. No significant differences between the candidate and adaptive interfaces were found.

2.2 Adaptive multi-sensory displays in simulated flight

Tannen (2000) assessed the effectiveness of adaptive multi-sensory displays for aiding target acquisition in an operationally relevant simulated flight task. HUDs and helmet-mounted displays offer some advantages for target detection scenarios. However, their utility is often constrained by characteristics unique to these technologies (e.g., narrow field of view, limited resolution, additional helmet weight, etc.). Tannen et al. (2000) proposed to compensate these limitations by the integration of spatial audio cues with standard HUD and head-coupled, helmet-mounted display symbology. The seven interfaces that were tested comprised combinations of adaptive and non-adaptive head-coupled visual and spatial audio displays designed to aid target acquisition. The visual cuing display consisted of a look-to-line and range indicator that was head coupled and projected onto the surface of the simulated flight environment.
The spatial audio display consisted of pulsed, broadband noise, displayed over a set of headphones, which appeared to emanate from the direction of the target.

In the non-adaptive cuing conditions, the visual and spatial audio cues were present throughout the entire flight trial whenever a target appeared in the field of regard. In contrast, in the adaptive conditions, the modality of the cuing interfaces was determined by the pilot's head orientation. For example, the adaptive visual display was activated when targets were within ±15º of the center of the pilot's head orientation. Conversely, the adaptive spatial audio cue was initiated when targets were greater than ±15º from the pilot's line of gaze. The pilots were asked to acquire ground and air targets while they followed a prescribed flight path and maintained a set airspeed and altitude.

An analysis of target acquisition performance indicated that all multi-sensory interface configurations enhanced performance relative to the standard non-cued display and the non-adaptive spatial auditory display. This effect was especially pronounced for ground targets. Moreover, multi-sensory displays, on average, were found to provide an 825-msec advantage over the non-adaptive visual cuing display for the designation of ground targets that were initially outside of the pilot's line of gaze. The advantages of multi-sensory displays were also reflected in pilots' overall ratings of perceived mental workload (National Aeronautics and Space Administration Task Load Index), which were found to be approximately 30 points lower than the standard non-cued and non-adaptive spatial audio displays.

2.3 Adaptive pilot-airplane interface

Mulgund and Zacharias (1996) presented an architecture of the adaptive pilot-airplane interface (PVI). The adaptive interface uses computational situation assessment models (based on Bayesian networks) and pilot workload metrics to drive the content, format, and modality of cockpit displays. The main purpose of the PVI concept is to support a tactical pilot's situation awareness and decision-making. The content, format, and modality of the adaptive pilot/vehicle interface are controlled by PVI control module. The overall architecture of adaptive interface is presented in Figure 1. PVI control module is driven by two key information streams: 1) the *content path*, driven by a tactical situation assessment module that uses avionics system outputs and the pilot's information needs; and 2) the *format path*, which uses an estimate of the pilot's state (workload level, attentional focus, etc.) to determine the most appropriate content, modality and format for conveying the required information to the pilot.

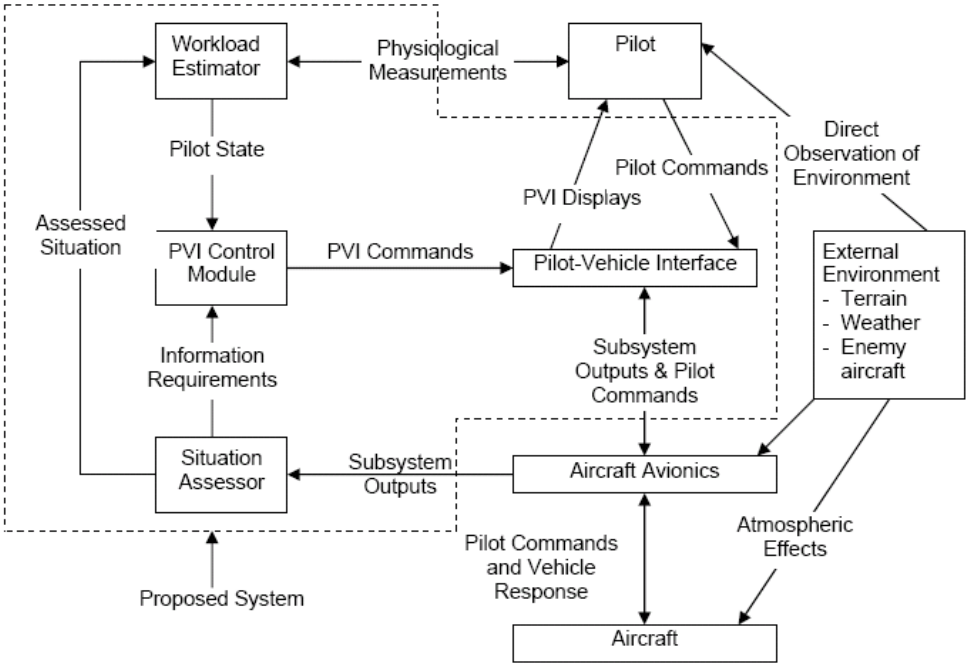


Figure 1 - Functional diagram of Situation-Driven Adaptive Interface (modified after Mulgund and Zacharias,1996).

The content path is based on the Crew/System Integration Model – that is integrated model of the air crew situation assessment and decision-making that has been using for the fighter attack mission and air superiority modeling (Mulgund, 1996). The *content path* consists of following stages:

1) Information processor module includes the following two elements: 1) the continuous state estimator that uses avionics system outputs to generate estimates of the aircraft's tactical situation (velocities, position, attitude, subsystems state, and states of the targets and threats); 2) discrete event detector that generates occurrence probabilities of mission relevant events (system failure, request for action, mission0related millstone)

2) Situation assessor block uses the estimated states and detected events to generates an assessed situation (S), which is a multidimensional vector defining the occurrence probabilities of the possible tactical situations that face the pilot.

A fixed and predefined set of situations is assumed, determined only by the mission relevancy. The situation assessor relies on Bayesian networks

3) Information filtering module: uses the given situation (S) to filtering the information stream to determine what information must be presented to the pilot to support his/her situation awareness (SA) and procedure execution. Filtering strategy relies on the hierarchy of the events, goals, and situations and a prioritization of information in relation to these (Endsley, 1992). The output of the module is the specification of the information presented to the pilot.

The *format path* consisting of following two stages: 1) the workload estimator, 2) The display configuration and adaptation strategy. The workload estimator includes:

a) Physiological processing system that uses such indices as: pilot pulse, respiration rate, eye blink rate, eye line of sight (HMD- mounted eye tracker), and EEG to compute physiological correlates of pilot workload.
b) Subjective and performance-based workload model, which provides the additional workload measures form off-line subjective evaluations and performance based assessment techniques. The individual on-line measures are fused together to aggregate indicators of pilot states

The display configuration and adaptation strategy (DCAS) uses the pilot state indicators and the pilot information requirements to determine how to configure the PVI displays. Implementation of the DCAS in the form of an expert system will use two principal knowledge bases (KB):

a) Display configuration KB contains the specifications of all normal display modes, formats and contents. The KB defines the baseline no-adaptive PVI, that may be manipulated by pilot by switches.
b) Human performance model KB contains model based on the principles of the human perceptual, cognitive, and response capabilities. This model provides a rule-based guidance how to adapt the PVI to a given situation. The output will appear on the head-down, head-up or helmet mounted displays. Auditory cueing could take form of synthesized speech alerts, warning tones, or 3-D localized sounds.

2.4 Adaptive interface for terrain navigation

Baus et al. (2002) developed hybrid navigation system that adapts the presentation of route directions to different output devices and modalities. The system takes into account the varying accuracy of positional information according to the technical resources available in the current situation. This system also adapts the information presentation to the limitation of user cognitive resources. This resource-adaptive navigation system (project REAL) consists of three major components. First, an information booth that consists of a 3Dgraphics workstation, where a virtual walk-through through the environment is shown by a virtual presenter, uses complementary spatial utterances and meta-graphics. Second, an indoor navigation system has been build based on strong infrared transmitters mounted at the ceiling and small PDAs as presentation devices. These are used to display simple sketches of the environment received via infrared method. The third component is an outdoor navigation system that uses a small laptop in combination with a head mounted display. A GPS system determines the user's actual position and an electronic compass tracks the user's orientation.

A single 3Dmodel of the environment is used to produce walkthroughs at the information booth and sketches for the mobile use. Adaptation services include the choice of camera perspective and path as well as the decision to include landmarks and interactive areas in the graphics. The REAL system tailors the presentations to a variety of technical limitations. Besides the size, resolution and color capability of the display, the system takes into account the computational power of the used device (information booth, PDA, and wearable computer). A specialty of REAL is the ability to integrate two different approaches to location sensitivity: active and passive location sensitivity. The system considers a variety of parameters that affect the cognitive resources, i.e. the walking speed, spatial familiarity and time pressure. For the navigation in buildings the IRREAL subcomponent was developed. IRREAL transmits interactive texts and graphics, very much like hypertext documents. This enables the user to interact with the presentation, although there is no bi-directional connection. The generated presentations are arranged in a presentation tree consisting of nodes, which may contain texts or graphics.

Through the use of transmission probabilities assigned to the different parts of a presentation tree it is possible to adapt the presentation to the user's walking speed. If the user stays in a transmission area for a short time the device will receive only the information with high priority, e.g., graphical walking directions. The more time the user spends in a transmitting area more complex the information about the environment will become available. In the ARREAL project a navigation system for pedestrians in an outdoor scenario was developed. ARREAL consists of four components: A sub-notebook, used for the relevant computations. For graphical or textual output a special clip-on for glasses is used. The users' position and orientation in the environment is determined through the use of a small GPS and a magnetic tracker. The magnetic tracker was modified and equipped with two additional buttons, so that it

can be used to interact with the system analogously to a standard two-button computer mouse. The modified tracker is used as a 3Dpointing device, e.g., the user can retrieve additional information by pointing on a building. On the small clip-on display (640x320 Pixel) sketch-like graphics are shown from birds-eye- or egocentric-perspective.

Overview maps are used to visualize the user's current position in the environment. Graphics from the ego-perspective view are used to present more detailed information about the environment, e.g., information about buildings in the current line of sight. In addition the system supports different levels of detail in the visualization. The system is able to visualize different portions of a map while changing from an overview to a detailed view of the environment. On the other hand textual or graphical annotations can be inserted, such as the names of streets or buildings. Navigational instructions are given by means of arrows that indicate turns to the user. System chooses between two modes: a birds-eye and ego-perspective. The ego-perspective is chosen when the system has adequate positional and orientational information. In cases were positional and orientational information is of inferior quality, ARREAL prefers the birds-eye perspective to the ego perspective. If bird’s eye-perspective is chosen, the precision of the positional information is encoded by the gray dots, resulting in a close-up of that area of the building. But in order to align the map to the walking direction, the system has to ensure the users correct orientation. The system also takes into account the user's current walking speed. If user moves fast, the system presents a greater portion of the map in order to help the user in orientation and at the same time to reduce the amount of information about buildings at the edges of the display. Since textual annotations at the edges of the display serve as menu items, the system reduces also the possibility to interact with the system.

3. ADAPTIVE INTERFACE FOR SUBMARINE SYSTEMS

Soulard (1992) presented an adaptive interface for submarine warfare system SAITeR (*Séquencement d'Activités Intelligent en Temps Réel, i.e. Intelligent Process Scheduling*). This application was designed and developed at the Advanced Research Department of TS.ASM Arcueil. SAITeR automatically performs a complete scheduling of the *Target Motion Analysis* (TMA). Each task runs a specific data processing algorithm whose triggering depends on operational and technical context evolution. SAITeR consists of two parts:

1) An automatic part (A) triggers algorithms depending on the operational context (township maneuvers, detected vessel maneuvers, the source of detection (mono- or multi-sensors detection, new contact or loss of detection), and results of the last algorithms.
2) A manual part (M) enables the human operator to trigger interactively particular algorithms on a small number of vessels in case of bad results from the automatic part (A).

The SAITeR controls the amount of information displayed. Analysis of the (A) part screen load (number of vessels and delay of presence) can lead to reduction of information displayed (e.g. the most threatened vessels or the vessels processed by the (M) part will be displayed). Moreover, the system reinitializes and updates the operator model by continuous analysis of human activities. The system takes into consideration some operator habits during performance of particular tasks. These individual human characteristics can be stored by the system in an operator model as yielding for simplification of the task.

Soulard (1992) suggested a diversification of interaction media to reduce visual information overload and improve human operator performance. The multimodal interface composed of following elements were proposed: 1) a touch entry screen (in place of some buttons), 2) a voice input to keep eyes on the screen during some commands, and 3) the speech synthesis under certain conditions, such as the use of headphone to reduce the ambient noise or the use of short messages. The generic architecture of adaptive interface is composed of three main modules:

1) *Media Management Module* that formats the events arriving from the different media or devices.

2) *Multimodal Request Understanding Module* that manages the multimodal request from the operator. Based on a linguistic and semantic analysis of the formatted events from the media manager, this module provides requests that are syntactically and semantically correct to the upper module.

3) *Dialog Understanding Module* that controls the dialog consistency i.e. when the operator makes a multimodal request of: 1) finding the current task of the operator, 2) dynamic updating of the task model, the operator model and the interactions history by analyzing the interactions, and 3) managing the strategy of the system and at anticipating the further task.

4. ACTIVE USER COLLABORATIVE INTERFACES

4.1 Adaptive interface for control of mental workload

Saiwaki (1996) described the adaptive interface that controls the level of the mental task difficulty according to the user's mental condition. The system measures and analyzes several physiological indices of the user completing the audio-visual mental task presented on the display. Than, it deduces the concentration and emotional tension level of the user, based on the extracted specific features of the physiological indices. Finally, the system adjusts the control parameters of the task to the user concentration and tension level. The system is composed of 3 stages:

1) EEG, ECG, and changing rate of SPR, are measured as original biological signals and physiological indices are extracted by biological signal processing. The following indices are used: heart rate (HR), and respiratory sinus arrhythmia (RSA); changing rate of SPR; distribution of EEG's peak frequency. 2) The level of emotional tension and concentration of the user are estimated from indices. The system learns the relations between user's mental conditions and the indices by pre-experiments in advance. The neural network is utilized for learning of these relations. 3) Mental task is controlled on the basis of the concentration and tension level assessed in the previous stage. The level of task is changed by adaptation of control parameters of the task, the picture size, color, moving speed, and sound tone.

4.2 Active user-interface paradigm

Brown and Santos (1999) developed an active user interface for PESKI system. The PESKI system (*Probabilities, Expert Systems, Knowledge, and Inference*) is an integrated probabilistic knowledge-based expert system development environment utilizing Bayesian Knowledge-Bases as its knowledge representation. PESKI provides users with engineering agents for knowledge acquisition, verification and validation, data mining, and inference, each capable of operating in various communication modes to the user. Authors claimed that active user interfaces serve as actuators in the human-machine interface, and allow the user to interact with the computer in a naturalistic/symbiotic manner. The active interfaces are capable of multi-levels of collaboration and autonomy. The user of an active user interface is fully aware of any actions, whether explicit (authorized consent) or implicit (implied consent), taken by the interface and has a complete, intuitive understanding of such actions. Brown and Santos (1999) developed for PESKI system intelligent knowledge engineering tools (agents) and integrated them using the active user interfaces paradigm.

PESKI consists of four major components (see Figure 2 or PESKI architecture):

. • *Intelligent Interface Agent:* translates English questions into inference queries and translates the analyses/inference results back into English, ; allowing intelligent communication exchange between the user and the system; *Inference Engine* ; includes intelligent strategies for controlling the selection and application of various inference engine algorithms (e.g. A*, 0), integer linear programming (ILP), genetic algorithms (GAs) to obtain conclusions to user queries,

- *Explanation & Interpretation* module; keeps track of the reasoning paths the inference engine; allows the user to query the system about how and why an answer was derived.
- *Knowledge Acquisition & Maintenance*; automatically incorporates new or updated expert knowledge into the knowledge base.

The active user interfaces paradigm was used to organize the PESKI into three subsystems. The four above components serve multiple functions and each PESKI subsystem combines different components together for that subsystem. The *User Interface* is composed of the Intelligent Interface and the *Explanation & Interpretation* components, as well as the interface components for the various engineering agents. The *Knowledge Organization & Validation* consists of the *Explanation and Interpretation* component along with the human expert and knowledge engineering tools.

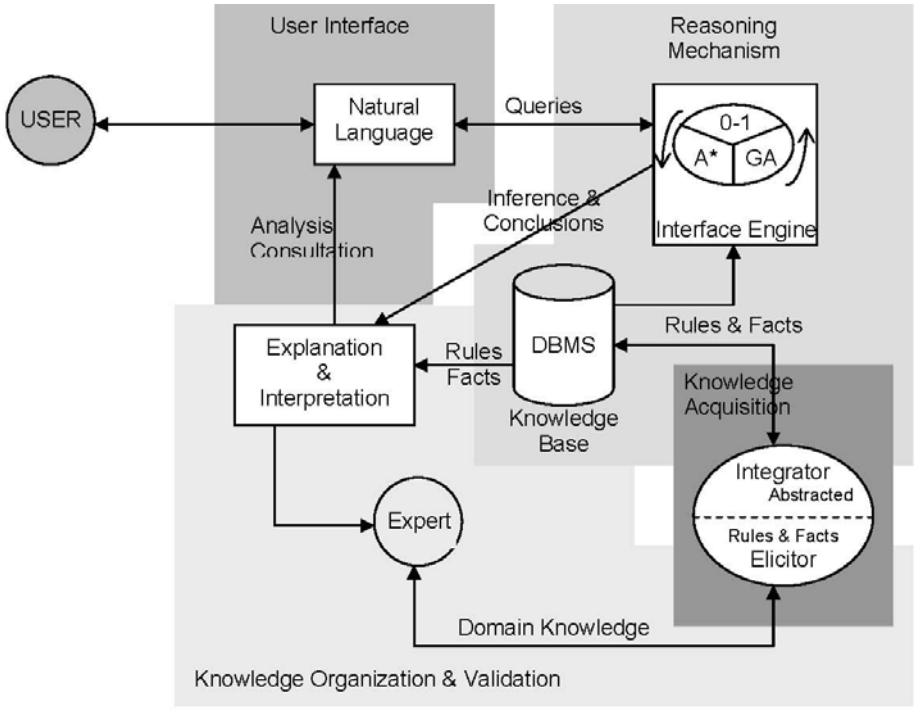


Figure 2 – PESKI Architecture (after Santos, 1999)

Organization is accomplished by communicating with the *Knowledge Acquisition & Maintenance* component, ensuring compliance with the BKB consistency constraints. The *Reasoning Mechanism* consists of the *Inference Engine* and the *Knowledge Acquisition & Maintenance* components. Assistance was provided by developing and maintaining cognitive model of the user. The user model captures the goals and needs of the user within the PESKI environment, as well as possible system events that occur, within a probabilistic representation/model of the PESKI environment. The interface agent determines the how, when, what, and why to offer assistance to the user. The agent is capable of offering assistance for such goals as which agent to use to correct a BKB consistency constraint violation as well as suggesting the user preferred communication mode for a given agent.

The *Knowledge Acquisition & Verification* is achieved through the MACK agent, who automatically and incrementally confirm consistency of the knowledge elicited from the expert and provides assistance by identifying the source of any inconsistency and proactively suggesting corrections. Regular incremental checks preserve both probabilistic validity and logical consistency as knowledge is acquired presumably under the expert's current consideration. PESKI's validation is performed using two agents - BVAL and GIT. BVAL validates a knowledge base against its requirements using a test case-based approach. Under certain conditions, the

knowledge base is corrected automatically via reinforcement learning of the probabilities. The graphical incompleteness tool (GIT) is used to visualize the knowledge base incompleteness for the user and actively provides solutions to correct it. The agent uses data visualization of the BKB and guides the user via color-coded shadings on how to repair the problem. The *Inference Engine* uses a performance metric-based approach to intelligently control a number of possible anytime and anywhere inferencing algorithms (e.g., A*, genetic algorithms). The control is specific to the given knowledge base and test case provided by the expert. Results are returned to the user via the *Explanation & Interpretation* subsystem of PESKI as they become available.

5. BRAIN-BASED ADAPTIVE COMPUTER INTERFACES

5.1 Asynchronous Adaptive Brain Interface

Millán and Mouri no (2003) developed an asynchronous *Adaptive Brain Interface* in which the subject makes self-paced decisions concerning switching from one mental task to another. This portable *Adaptive Brain Interface* (ABI) is based on the on-line analysis of spontaneous electroencephalogram (EEG) signals measured with eight scalp electrodes and able to recognize three mental tasks. This approach relies on an asynchronous protocol where the subject decides voluntarily when to switch between mental tasks. The simple local neural classifier is used to recognize (every 0.5 s) the mental task on which the subject is concentrating. ABI was used to operate two brain-actuated devices: a virtual keyboard and a mobile robot (emulating a motorized wheelchair).

The *brain computer interface* (BCI) is based on the analysis of EEG signals associated with spontaneous mental activity. The analysis is concerned with local variations of EEG over several cortical areas that are related to different cognitive mental tasks such as imagination of movements, arithmetic operations, or language. The EEG patterns embedded in the continuous EEG signal and associated with different mental states was determined. The machine-learning techniques were used to train the classifier and follow a *mutual learning* process where the user and the brain interface are coupled and adapt to each other. This accelerates the training process. In the presence of feedback, subjects achieved good performance in just a few hours of training. ABI has a simple local neural classifier where every unit represents a prototype of one of the mental tasks to be recognized.

It was found that this local network performs better than more sophisticated approaches such as support vector machines and temporal-processing neural networks (TDNN and Elman-like). This performance was achieved by averaging the outputs of the network for eight consecutive EEG samples (and still yielding a global response every 0.5 s). Once trained, the response of the network for the arriving EEG sample is the task with the highest posterior probability, provided that it is above a given probability confidence threshold

(otherwise the response is classified as “unknown”). The posterior probability distribution is based on the Mahalanobis distance from the EEG sample to the different prototypes.

Several demonstrations were developed to illustrate the wide range of systems that can be linked to ABI. The brain interface can be used to select letters from a virtual keyboard on a computer screen and to write a message. Initially, the whole keyboard (26 English letters plus the space to separate words, for a total of 27 symbols organized in a matrix of three rows by nine columns) is divided in three blocks, each associated to one of the mental tasks. The association between blocks and mental tasks is indicated by the same colors as during the training phase. Each block contains an equal number of symbols, namely nine at this first level (three rows by three columns). Then, once the neural classifier recognizes the block on which the subject is concentrating, this block is split in three smaller blocks, each having three symbols this time (one row). As one of these second-level blocks is selected (the neural classifier recognizes the corresponding mental task), it is again split in three parts. At this third and final level, each block contains one single symbol. Finally, to select the desired symbol, the user concentrates in its associated mental task as indicated by the color of the symbol. This symbol goes to the message and the whole process starts over again. Thus, the process of writing a single letter requires three decision steps.

The EEG potentials were recorded at the eight standard fronto-centro-parietal locations: F3, F4, C3, Cz, C4, P3, Pz, and P4. The sampling rate is 128 Hz. The raw EEG potentials are first transformed by means of a surface Laplacian (SL) computed globally by means of a spherical spline of order. Then the Welch periodogram algorithms were used to estimate the power spectrum of each SL-transformed channel over the last second. EEG sample had 96 features (8 channels x 12 components each).

5.2 EEG-based interfaces

Pope, Bogart, and Bartolome (1995) examined the utility of EEG for adaptive automation technology. These researchers developed an adaptive system that uses a closed-loop procedure to adjust the mode of automation based on changes in the operator's EEG patterns. The closed-loop method was developed to determine optimal task allocation using an EEG-based index of engagement or arousal. The system uses a bio-cybernetic loop that is formed by changing levels of automation in response to changes in mental workload demands. Thus, an inverse relation exists between the level of automation in the tasks and the level of operator workload. The level of automation in the task set could be such that all, none, or a subset of the tasks could be automated. The task mix is modified in real time according to the operator's level of engagement. The system assigns additional tasks to the operator when the EEG reflects a reduction in task engagement. On the other hand, when the EEG indicates an increase in mental workload, a task

or set of tasks may be automated, reducing the demands on the operator. Thus, the feedback system should not reach a steady-state condition in which neither sustained rises nor sustained declines in the EEG are observed.

In this study participants performed the compensatory tracking task of the Multiple-Attribute Task (MAT) Battery. The MAT Battery primary display is composed of four separate task areas or windows, comprising the monitoring, tracking, communication and resource-management tasks. Each of these tasks in the MAT set is designed to be analogous to a task that crewmembers perform in flight management and each can be made either manual (subject must manage task) or automated (computer manages task). In the version of the MAT developed for these studies, the monitoring, communication and resource-management tasks remained in automatic mode, and the compensatory tracking task was performed by the subject when in manual mode and only monitored by the subject when in automatic mode. Pope et al. (1995) reported that three indexes--beta/alpha, beta/(alpha plus them), and alpha/alpha--were able to distinguish between the feedback conditions, but the best discriminator was the index, beta/(alpha plus theta).

Prinzel et al.. (2000) developed a closed-loop, biocybernetic system to test various psychophysiological measures for their use in adaptive automation. Specifically, were assessed the use of the EEG band ratio, beta/(alpha plus them) on the basis of behavioral, system, and physiological data gathered under negative and positive feedback controls. Furthermore, the study was designed to determine how different task loads impact adaptive task allocation and system regulation of task engagement and workload. Participants operated a modified version of the MAT Battery. The MAT Battery is composed of four separate task areas, or windows, constituting the monitoring, compensatory tracking, communication, and resource management tasks. These different tasks were designed to simulate activities that airplane crewmembers often perform during flight. Only the monitoring, compensatory tracking, and resource management tasks were used for this study. The functioning of the monitoring and resource management tasks was controlled by a script file that controlled the sequence and timing of the events in the tasks. The compensatory tracking task was cycled between manual and automatic modes

Tracking performance was found to be significantly better under the negative feedback condition than under the positive feedback condition. These results suggest that the closed-loop system can facilitate performance and complements the task allocation and psychophysiological data supporting the use of the system for adaptive task allocation. The results showed that more task allocations were made under the multiple task condition. Therefore, the system appears to be sensitive to increases in task load. Participants also rated workload higher and performed the tracking task more poorly under the high workload condition. The EEG engagement index, however, was not found to discriminate between these two task conditions, although the value of the index was higher under the multiple task

condition than under the single task condition. Nevertheless, these results support that the single and multiple task conditions provided different levels of task load.

5.3 Interfaces with on-line self adaptivity

Vico et al. (2001) proposed to achieve the on-line self-adaptivity of the human-computer interfaces by implementation of the basic principles of classical behavior conditioning to the neural networks. This type of interface adapts without any a priori information of their interaction with the user. The prototype adaptive interface was developed to demonstrate the applicability of this learning technique to the adaptation of user interfaces. Classical conditioning deals with unconditioned stimulus (UCS) that automatically elicits an unconditioned response (UCR). If some given conditioned stimuli (CS) precede another UCS that elicits a concrete response, this CS will be associated with the UCR. This CS–UCS relation transforms in a conditioned responses (CR) that involves the specific generation of the UCR by the CS. The Sutton and Barto (SB model) model of the classical conditioning (that considers the temporal appearance of the UCS and CS) were implemented to the neural networks.

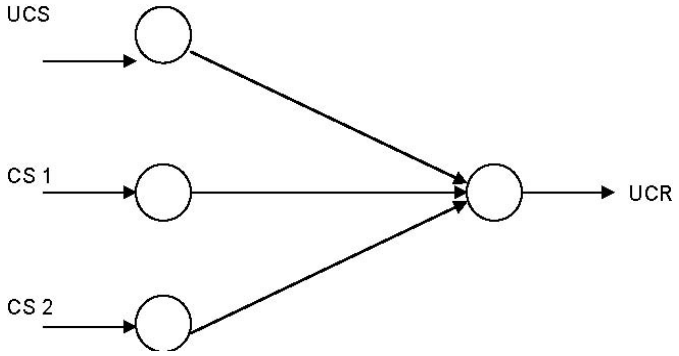


Figure 3 - Basic neural network architecture for the SB model (after Vico et al. 2001).

Adjustment of the synaptic weights between neurons was made in an incremental learning fashion. This adjustment is done according to the following rule:

$$\Delta W_{ij} = \alpha(y - \bar{y})\bar{x}_i$$

With

W_{ij} = synaptic weight(or association level between stimuli and response)

$\bar{x}_i$ = temporal trace of the CS

$\bar{y}$ = response level(UCR or CR)

y = trace of the response

Both traces implement a short-term memory of recent activation levels, and are computed according to the following equations:

$$\bar{x}_i(t+1) = \beta \bar{x}_i(t) + x_j(t)$$
$$\bar{y}(t+1) = \lambda \bar{y}(t) + (1-\lambda) y(t)$$

Where $x(t)$ represents the *over time* CS, $y(t)$ is the response level at time t, and β and λ are constants related to the size of the temporal integration window.

The model works by increasing a weight when a CS comes before the arrival of the UCS and decreasing it if the predicted UCS does not arrive. In order to avoid recurrent self-connections and overall inhibitions, the CS is `artificially' maintained up to the arrival of the UCS to get the memory traces necessary for associating both stimuli. The neural circuit shown in Fig. 4 constitutes the building block of a network that learns temporal relations between stimuli and responses. The particular architecture of the network must account for all the input–output relations that might be present in the interface behavior.

The implemented prototype is windows-based application that allows the user to build sentences from limited sets of words. These words are grouped in three different classes: pronouns, verbs, and objects, and can be extracted from menus that can be opened up by clicking on the button labeled with the corresponding class identifier. Finally, the `OK' button restarts the system, allowing a new sentence to be typed. The structure of the neural system used in the adaptive interface is presented in Fig. 4. The basic circuit of Fig.3 is expanded to implement all possible combinations of events and actions. This two-layer network has an input layer that stores the user-generated events and an output layer that produces actions. After the group of user feed the system a set of events can be grouped, according to their nature. Two classes of events sets were obtained: user's commands (environmental stimuli perceived by the interface) and internal actions (interface's responses) that have precise consequences on the computer system.

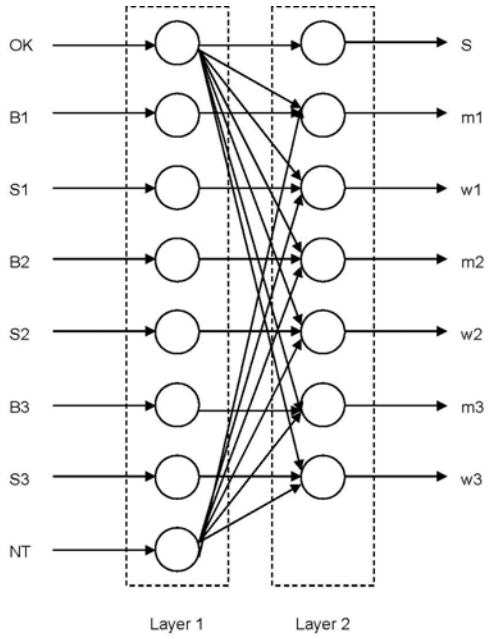


Figure 4 – Overall network architecture (Vico, 2001). User's commands and internal actions

The command–action relations are represented by UCS–UCR excitatory connections (positive weights). Stimuli arrive to the network from the left, generating responses through the excitatory rigid connections represented by small solid circles. The model neuron is defined as a summation of incoming activity. A neuron remains in a resting state (output to 0) if there is not enough activity to activate it, and outputs a maximum value (1) when overall activity exceeds the threshold. Rigid excitatory connections are adjusted in such a way that the activation of the pre-synaptic neuron is able to elicit a post-synaptic response. As the user enters sentences, the interface trains itself, and some connections start changing their initial values. At some point, the interface starts eliciting CRs (anticipating user's commands). This unexpected behavior might or might not fit with the user's interests. The sequence of actions that follows a CR tells the system whether this command was or not appropriate: if the user feeds an event that keeps the expected sequence on the track then the acquisition is effective, and in the future this event will be automatically generated by the interface while the sequence remains valid. If, instead, the user is forced to go back, giving actions that break the expected sequence of commands, then the interface has to reconsider its CR, extinguishing this behavior in the future.

This interaction between the user and the interface takes, as a consequence, the initial random configuration of the network to a stable state where the interface performs the predictions and elicits adequate actions to facilitate the task. This behavior will be stable as long as the user's commands always follow the same series. If the user changes the sequence, then the interface is

taken to a different state where some previous actions are extinguished and some new skills are learnt. This interface can be upgraded to a sort of adaptive system that predicts the arrival of a user's commands and, furthermore, performs appropriate actions that speed up the interaction between the user and the interface.

The main difference between the proposed approach and traditional methods is that the neural network by itself rules the way the interface operates. While most intelligent skills are pre-included in the user interface, the network introduces non-modifiable connections to implement the pre-wired reactions (the interface itself) and modifiable connections that account for all possible associations among user actions and interface behavior. Initially, this method applies to non-modal interfaces, in which system response to one event depends only upon the event. However, the learning mechanism underlying this technique converts the original non-modal interface in a modal interface where the system response to one event is related to previous event by means of the memory traces stored in synaptic weights of the neural network.

6. INTERFACES FOR ADAPTIVE CONTROL SYSTEMS

Adaptive control is an active and diverse research area with many different applications. An adaptive control system can be defined as a feedback control system intelligent enough to adjust its characteristics in a changing environment so as to operate in an optimal manner according to some specified criteria (Wahi et al., 2001). Review of literature shows that adaptive control systems have achieved great success in aircraft, missile, and spacecraft, and process control applications. Applications of adaptive control can be broadly divided into application of classical and intelligent control techniques. This literature review focuses on the intelligent control techniques that combine and extend theories and methods mainly from artificial intelligent area such as, neural networks, fuzzy logic, and evolutionary programming. These computing techniques are used individually or in combinations.

6.1 Fuzzy Logic Applications

This section discusses recent application of fuzzy logic in adaptive control systems in aircraft, and system and process related control applications.

Fuzzy logic based fight control system

Zhou et al. (1997) proposed for a fuzzy logic based fight control system for a hypersonic transporter in order to provide the longitudinal stability in the hypersonic region and to improve the response of the vehicle as well as to make the response exactly follow the commands. Fourteen fuzzy inference rules were used to model human operator behavior and max-min composition algorithm was used in the inference model. The model was used at four flight points of the flight envelope. The evaluation included of the following: 1) response to the hypersonic transporter with the fuzzy logic controller to an initial disturbance of the angle of attack in the hypersonic region, in which the vehicle without the fuzzy logic controller was dynamically unstable, 2) comparison of the fuzzy logic controller with a conventional stability augmentation system, and 3) robustness of the fuzzy logic controller to flight condition variation. Figures 5.1.1.1 and 2 shows general construction of a fuzzy logic controller and functional block diagram of FLC of the hypersonic aircraft

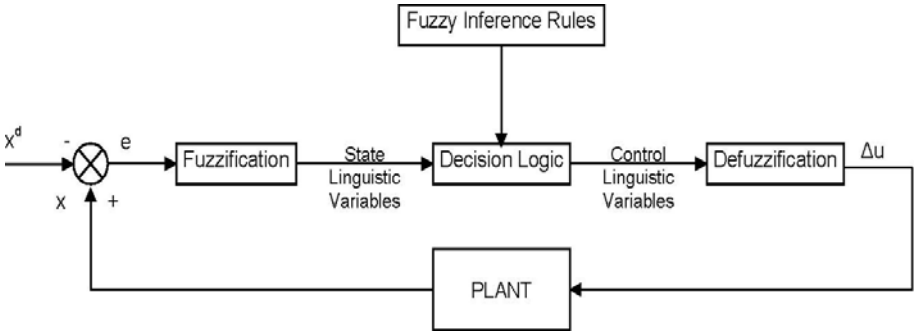


Figure 5 - General construction of a fuzzy logic controller (after Zhou et al. 1997)

The results showed that the fuzzy logic controller had the ability to stabilize the vehicle in the hypersonic region, and was fairly robust across the flight envelope. The authors also found that the fuzzy logic controller may be more capable than the conventional stability augmentation system.

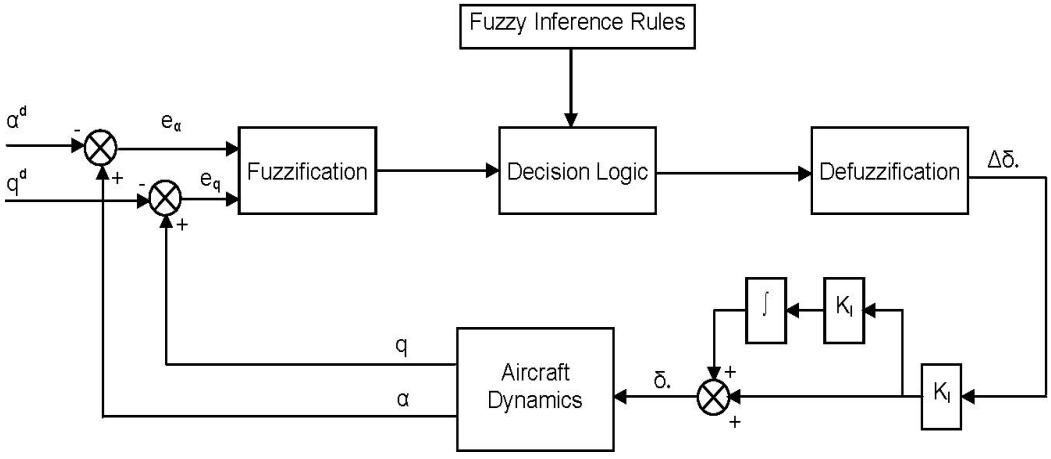


Figure 6 - Functional block diagram of fuzzy logic controller for hypersonic aircraft (after Zhou et al. 1997).

Multiple fuzzy controllers for flight control systems

Schram et al. (1997), in their study, implemented multiple fuzzy controllers for anticipating failure of flight control systems using a fuzzy logic expert system. In this study, the rule-based system was used as outer loop controller and additional supervisory rules were defined in case of failures. These ascertained the achievement of smooth and fast switching between different control modes in the same framework. Using fuzzy sets and fuzzy logic operations, the study designed a fuzzy reasoning system that acted as a controller. Figure 7 shows the structure of a typical fuzzy logic controller.

The control strategy was stored in the form of IF-THEN rules in the rule base. These rules represented a static mapping from inputs (measurements) to outputs (control actions). Dynamic filters were used to introduce dynamics (error and derivative of error) and integration of the output. The membership functions provided a smooth interface to the numerical process variables. The *fuzzification* module determined the membership degree of the antecedent fuzzy sets. The inference mechanism combined this information with the rule base and determined the output of the rule-based system. In order to obtain a non-fuzzy signal, the output in the form of a fuzzy set was *defuzzified*. The aggregation and *defuzzification* phase were then combined in one step by the weighted fuzzy-mean method.

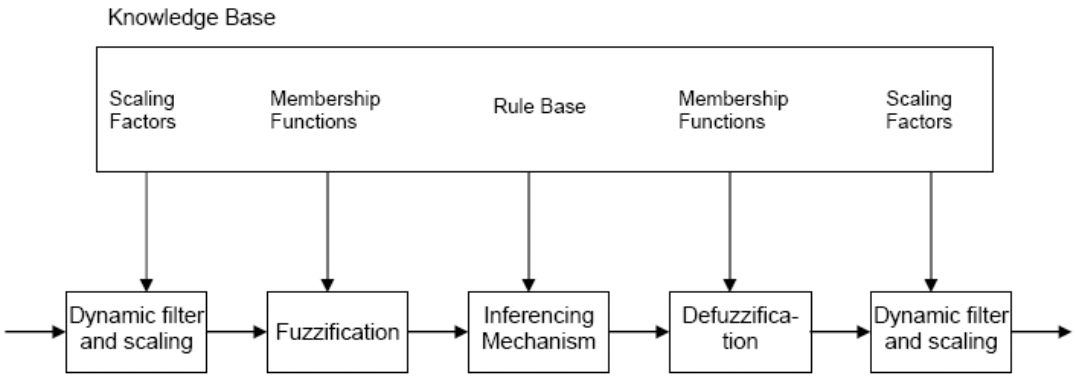


Figure 7 - Block diagram of the fuzzy logic controller (after Schram et al. 1997)

Flight control and mission planning for unmanned aerial vehicles

Vachtsevanos et al. (1997) proposed a hybrid hardware-software platform to support flight control and mission planning algorithms for an autonomous unmanned aerial vehicle. The objectives of the project were to demonstrate automation technologies for vertical takeoff and

landing and to develop an integrated product and process development. The autonomous unmanned vehicle configuration consisted of a mission planner that included a supervisory controller and fuzzy route planner, fault tolerance, and navigator. The configuration also used a fuzzy flight controller that used phase portrait assignment algorithm. This algorithm is capable to utilize experimental data or simple nonlinear system models and heuristic evidence to arrive at the phase plane or phase space representation. Figure 8 shows the ASRT configuration

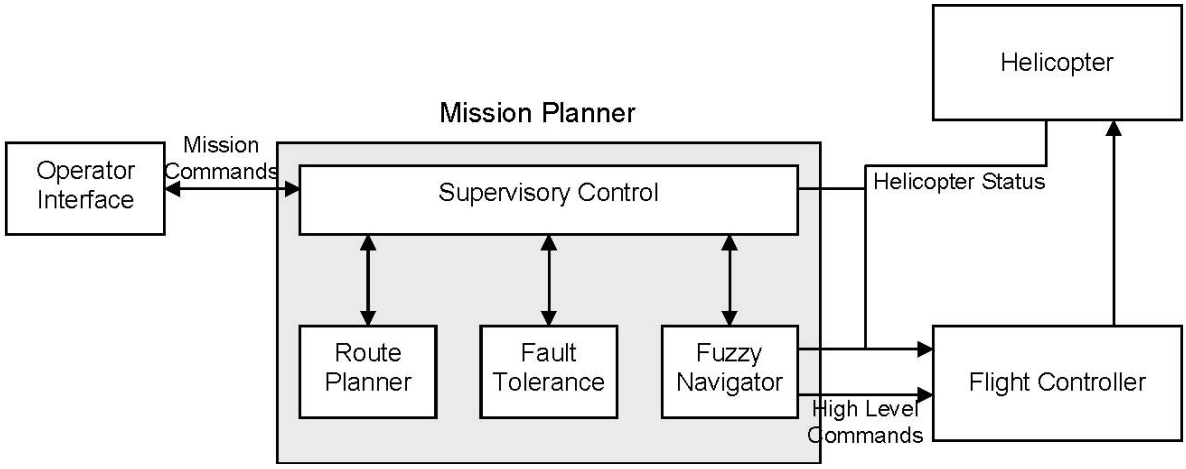


Figure 8 - The ASRT configuration (after Vachtsevanos et al. 1997).

In this ASRT model, the high-level supervisory controller provided the start and destination points to the route planner. The route planner's task was to generate the 'best' route in the form of waypoints for the helicopter to follow. It used a modified A* search algorithm that minimized a suitable cost function consisting of the weighted sum of distance, hazard, and maneuverability measures. The cost elements were expressed as fuzzy membership functions. Figure 9 shows the flowchart of the route planner. A fuzzy navigator was designed to command the helicopter in the navigation mode. The

Adaptive human-computer interfaces vehicle followed a series of waypoints in a intelligent manner in order to achieve the best compromise between waypoint spatial compliance and energy management.

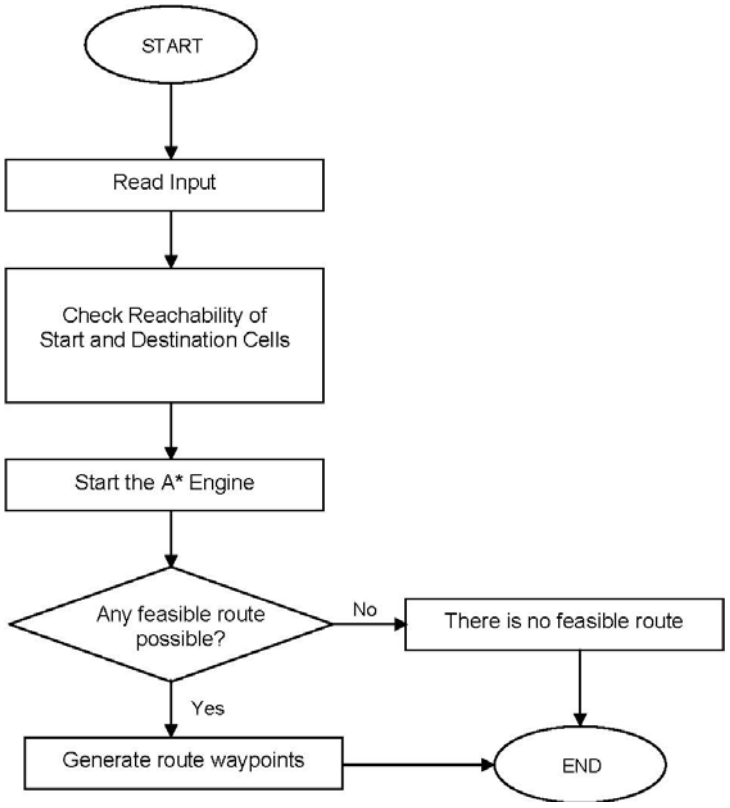


Figure 9 - Flow chart of the route planner (after Vachtsevanos et al. 1997).

The model also consisted of a fuzzy fault tolerance module where critical sensor or component failure modes were stored in a fuzzy rule base as templates. Real time sensor data were then fuzzified and an inference engine was employed to compare the incoming signals with the stored information. In the model, the failure detection and identification architecture entailed both online and offline learning algorithms and also means to associate a degree of certainty to the decision making process. The ASTR model consisted of a fuzzy flight controller that utilized phase portrait assignment algorithm (PPAA). Preliminary results showed that introduction of fuzzy logic based algorithms for the flight control and mission planning, in conjunction with other decision support tools, offers promise that such autonomous vehicles can accomplish a true mission. The authors, however, emphasized for further studies in order to achieve real autonomy in terms of intelligent attributes-adaptation, learning and fault-tolerance.

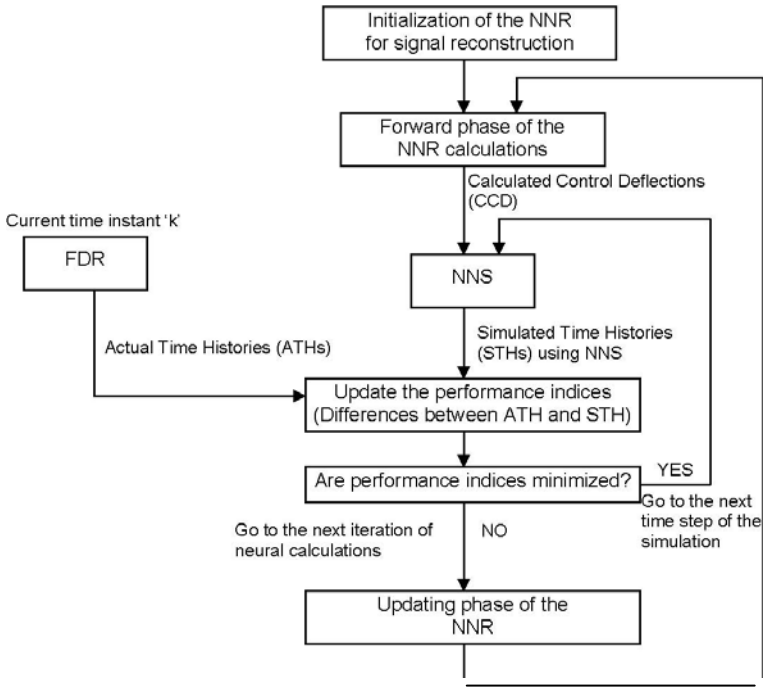
Virtual sensors in flight control systems

Oosterom and Babuska (2000) developed and implemented a virtual sensor for normal

acceleration in flight control system that was used in a small commercial aircraft. The consolidated outputs of dissimilar sensor signals were used as the inputs of the virtual sensors. The application of virtual sensors in flight control systems makes it possible to distinguish between two real sensors in the case of a failure and therefore either increase the safety of the system or reduce the cost of the system. A Takagi-Sugeno (TS) type fuzzy model was utilized for this purpose. The results showed that variance accounted for (VAF) index was higher in TS model compared to the linear model and that root mean squared error was less in TS model compared to the linear model. The authors proposed for future studies to investigate the robustness of the virtual sensors with respect to variations in the aircraft weight and the center of gravity and also in 'pilot in the loop' simulations.

Virtual flight data recorder

Napolitano et al. (1999) utilized neural network and fuzzy logic for the development of a virtual flight data recorder on commercial airliners. In their study, a neural network simulator (NNS) was used to predict the aircraft control surface deflections by using neural network or fuzzy logic reconstructor (NNR or FLR). Figure 10 shows the block diagram of the model.



NNR = Neural Network Reconstructor

NNS = Neural Network simulator

FDR = Flight Data Recorder

Figure 10 - Block diagram of the virtual flight data recorder (after Napolitano et al. 1999).

The NNS was trained off-line, using available flight data for the particular aircraft. Then the NNS was interface with the NNR or FLR. The outputs of the two reconstructors are the control surface deflections that minimize a performance index based on the differences between the available data from the flight data recorder and the output from the NNS. The results for the study showed that both schemes provide accurate reconstructions of the control surface deflections time histories.

Intelligent flight support system

Burdun and Parfentyev (1999) investigated the problem of intelligent flight support under complex operational conditions. In this study, a ‘chain reaction' mechanism of a flight accident was described. An affordable method of flight safety enhancement in advanced aircraft was suggested. The method employed the concept of a hybrid intelligent pilot model, which combined positive anthropomorphic and mathematical properties. A central component of this artificial intelligence model was a comprehensive knowledge base in the form of fuzzy situational tree-network (FSTN) of flight.

A conceptual framework and some algorithmic issues of the method were discussed. Examples of FSTN prototyping were described in the article. Potential applications included an intelligent pilot-vehicle interface, automatic flight-envelope protection, autonomous (robotic) flight including multiple vehicle systems, resolution of conflicts in close free-flight air space, and others.

Active control of aircraft dynamics

Jeram and Prasad (2003) designed an active control system that alters the force-feel characteristics of a two active-axis-sidestick during adverse aircraft-pilot coupling (APC) events to provide a tactile avoidance cue. These events, also called *Pilot Induced Oscillations* (PIO), typically occur when the total aircraft dynamics unexpectedly deviate from the pilot’s expectations of control and response. This is often due to nonlinear effects such as rate limiting elements that make the aircraft dynamical response sluggish. In this study, a fuzzy logic based PIO detector was used to estimate the dominant frequency, phase lag, and actuator rate limit, and triggers a tactile avoidance cue that uses friction, radius of motion, and bobweight dynamics to communicate the dynamical nature of the aircraft that precipitates a PIO event. Preprocessing of PIO detection is shown in Figure 11.

The PIO tactile avoidance cues presented in this study explored three new elements for carefree maneuver systems: 1) They apply to a controllability limit rather than a structural limit, 2) They use a logic based detector rather than an arithmetic cue detector, and 3) The

tactile interface uses radius of motion, friction, and force-feel dynamics rather than displacement based force cues.

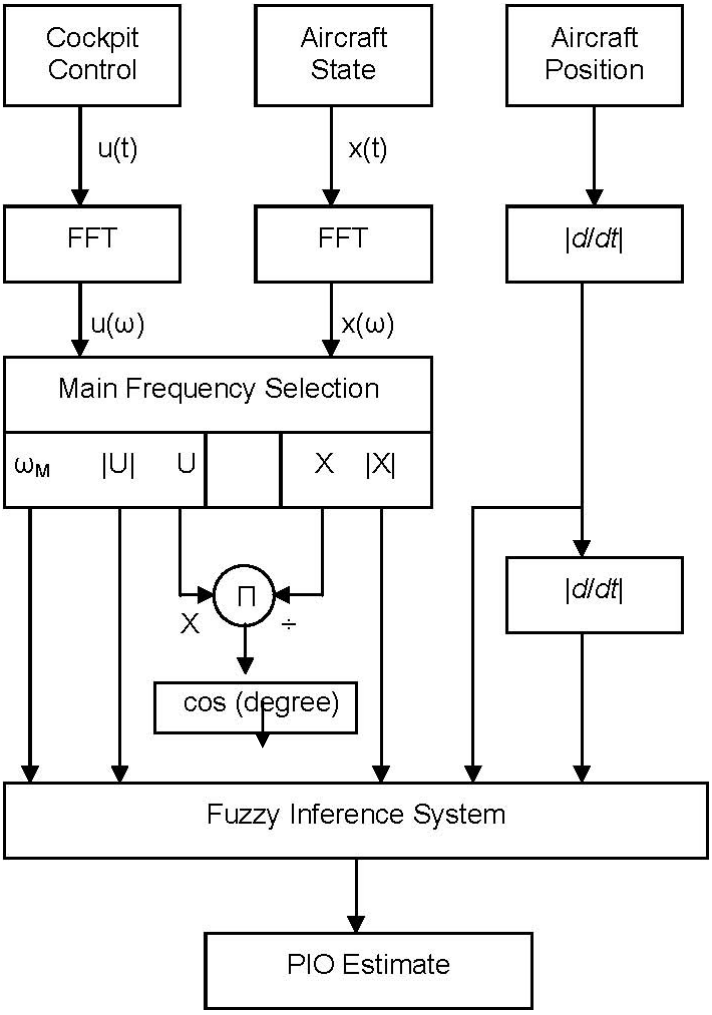


Figure 11 - Preprocessing of pilot induced oscillation (PIO) (after Jeram and Prasad, 2003).

The study found that a unidirectional friction force up to 40% of the maximum static deflection force could provide an effective, intuitive tactile cue that the pilot's stick movement exceeded some rate limitation within the total aircraft. This saturation cue was effective when there was a fundamental directional relationship between the rate limited element and the inceptor movement. However, the authors concluded that, It may not be appropriate for aircraft with unstable aerodynamics requiring multiple control surface actuator reversals during a maneuver. It was also found that the range of motion (RoM) cue was marginally useful. It can help control PIO events, but it does so at the cost of reduced pilot control authority. Also, some variations of this cue, where force gradient is altered, can produce objectionable interference with pilot commands. The authors suggested that similar PIO countermeasures may be implemented by the flight control

system without the use of active cues.

On-Line Intelligent Processor for Situation Assessment

Mulgund et al. (1997) assessed the feasibility of developing an concept prototype for an *On-Line Intelligent Processor for Situation Assessment* (OLIPSA), to serve as a central processor to manage sensors, drive decision-aids, and adapt pilot/vehicle interfaces in the next-generation military cockpit. The approach integrates several enabling technologies to perform the three essential functions of real-time situation assessment: 1) Event detection uses a fuzzy logic processor and an event rule base to transform fused sensor data into situational-relevance semantic variables, 2) Current situation assessment is performed using a belief network (BN) model to combine detected events into a holistic "picture" of the current situation, for probabilistic reasoning in the presence of uncertainty, and 3) Future situation prediction is carried out via case-based reasoning, to project the current situation into the future via experience-based outcome prediction. OLIPSA's performance was demonstrated initially in the defensive reaction portion of an air-to-ground attack mission, in which a pilot must deal with an attack from threat aircraft. Situation awareness models were developed to support the pilot's assessment of the threat posed by detected aircraft.

Conflict free flight path guidance system

Rong (2002) developed an agent-based hierarchical system that attempts to provide optimal and conflict free flight path guidance in situations where more than one type of conflict existed. An intelligent executive guidance agent, acting as a high-level arbitrator, received guidance information from lower-level weather agent and traffic agents. Figure 12 shows the overall architecture for agent based hierarchical system.

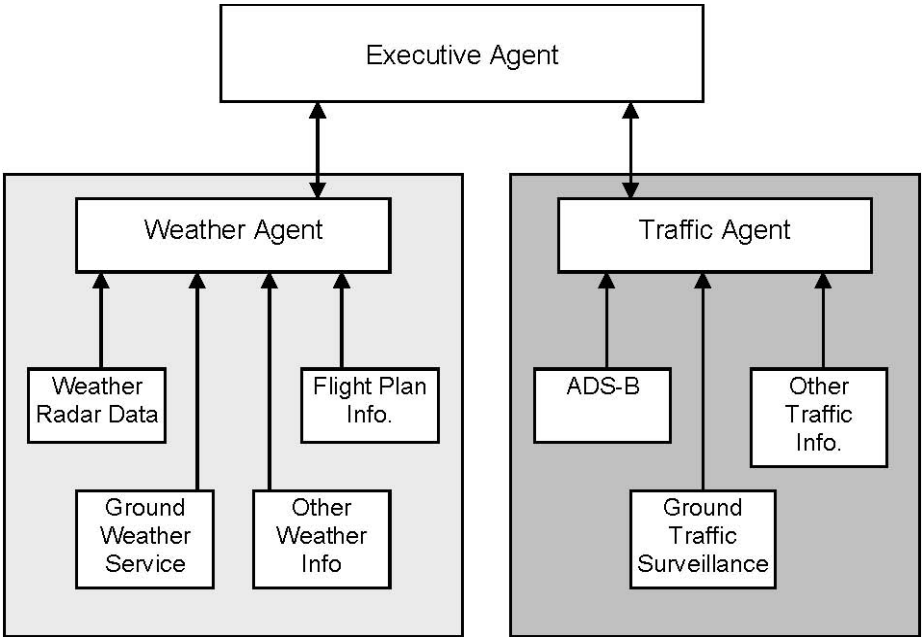


Figure 12 - Overall architecture of the agent based hierarchical system (after Rong (2002).

When the flight path guidance from the two agents conflicted, the executive agent arbitrated by considering the spatial and temporal characteristics of the conflicting guidance. It classified them as either tactical or strategic in nature, and then prioritized them according to a pre-defined rule base of conflict priorities. The arbitration function thus acted as a fuzzy controller, and gradually switched the guidance between the weather agent and traffic agent, providing conflict free flight path guidance, as the aircraft flied in and out of dangerous regions. Results of test cases presented in the paper demonstrated that the approach and algorithm could successfully resolve combined weather and traffic conflicts.

Intelligent and autonomous flight control system

Wu et al. (2003) investigated on an intelligent and autonomous flight control system for an atmospheric re-entry vehicle based on fuzzy logic control and aerodynamic inversion computation. A common PD-Mamdani fuzzy logic controller was designed for all the five re-entry flight regions characterized by different actuator configurations. A linear transformation to the controller inputs was applied to tune the controller performance for different flight regions while using the same fuzzy rule base and inference engine. An autonomous actuator allocation algorithm was developed, based on the aerodynamic inversion computation, to cover all the five actuator configurations with the same fuzzy logic controller.

Simulation tests were conducted to track both a benchmark trajectory and the nominal re-

entry trajectory. Test results showed that both the thrusters and body surfaces were able to conduct their roles in appropriate flight regions along the nominal trajectory. Tracking errors and the actuator usage were both well within their appropriate acceptable ranges. However, the bank reversals in the early part of nominal trajectory were too demanding for the thrusters, which revealed that there was a mismatch between the trajectory computation and the vehicle control regarding the thruster settings. Compared to the NLDI control approach, the proposed approach provided a better tracking performance, while having advantages in autonomous actuator allocation to guarantee the availability of the commanded control moments, and in handling non-linear actuator saturations (in both thrust and control flaps). Appendix A summarizes the reviewed studies on utilization of fuzzy logic controllers.

6.2 Neural network applications

Neural networks are known for their capabilities to approximate nonlinear mappings to a high degree of accuracy. Recently, neural networks have been widely used in the control of systems of transportation and wide variety of technological systems. Appendix B summarizes the reviewed studies on neural network based controllers.

Intelligent navigational aids

Caldwell et al. (1998) developed a neural network based landing approach navigation aid. The navigation aid provides the pilot with turning rate information that is based only on a non-directional beacon ground radio station and an automatic direction
Adaptive human-computer interfaces finder. Figure 13 shows the incorporation of phase identification algorithm into neural networks.

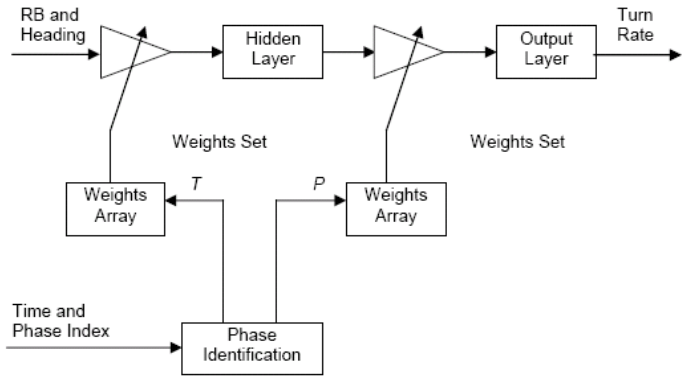


Figure 13 - Neural network and phase identification (after Caldwell et al. 1998).

The neural network controller determines a landing approach based on a seven-phase typical non-directional beacon system. In each phase, a feed forward neural network with one hidden layer with three nodes for non-directional beacon landing was used. A back-propagation learning strategy was used to determine the weights of the network. Simulation of eight cases showed that neural networks trained on human control patterns can be used as landing approach navigation aid.

Adaptive flight control system

Napolitano et al. (1999), in their study, demonstrated the capabilities of hardware based online learning parallel neural networks featuring neural schemes for fault-tolerant capabilities in a flight control system. Two different fault-tolerant schemes were introduced. The first scheme provided sensor failure detection, identification, and accommodation (SFDIA) for different kinds of sensor failures within a flight control system while the second provided actuator failure detection, identification, and accommodation (AFDIA) for different actuator failures. Simulation showed that by means of lower and upper bounds of auto and cross correlation functions, the controller was able to integrate AFDIA and SFDIA schemes without degrading performance in terms of false alarm and incorrect failure identification.

Near-optimal helicopter flight load synthesis

In their study, Manry et al. (1999) used neural networks for near optimal helicopter flight load synthesis (FLS) that is the process of estimating mechanical loads during helicopter flight, using cockpit measurements. First, modular neural networks were used to develop statistical signal models of the cockpit measurements as a function of the loads. Then Cramer-Rao maximum a-posteriori bounds on the mean squared error were calculated. Finally, multilayer perceptrons (MLP) for FLS were designed and trained that approximately attained the bounds or optimal performance. The authors, following the simulation, concluded that further studies need to be done to size the inverse networks in order to produce better bounds and to determine the objectivity of mappings directly from the training data.

A fault-tolerant flight controller design

Yan et al. (1999) applied minimal radial basis function neural networks called the *Minimal Resource Allocation Neural Networks* (MRAN) for fault-tolerant flight controller design. In their architecture, the MRANN controller aided the conventional controller.

The neural nets did not require off-line training and the scheme had good fault-tolerant

capabilities. The MRAN controller was illustrated for a fighter aircraft (F-8) longitudinal control in an autopilot mode for following velocity and pitch rate pilot commands under large parameter variations and sudden variations in actuator time constants. Results indicated that MRAN controller exhibited better performance than another feed forward inverse neural controller that used a gradient learning scheme.

Adaptive flight control system

Urnes et al. (2001), in their study, developed a damage adaptive flight control system that utilizes neural network technology to predict the stability and control parameters of the aircraft, and uses this data to continuously optimize the control system response. Figure 14 shows the block diagram of application of neural network by the IFCS design and the advanced flight controller to continuously optimize flight path response.

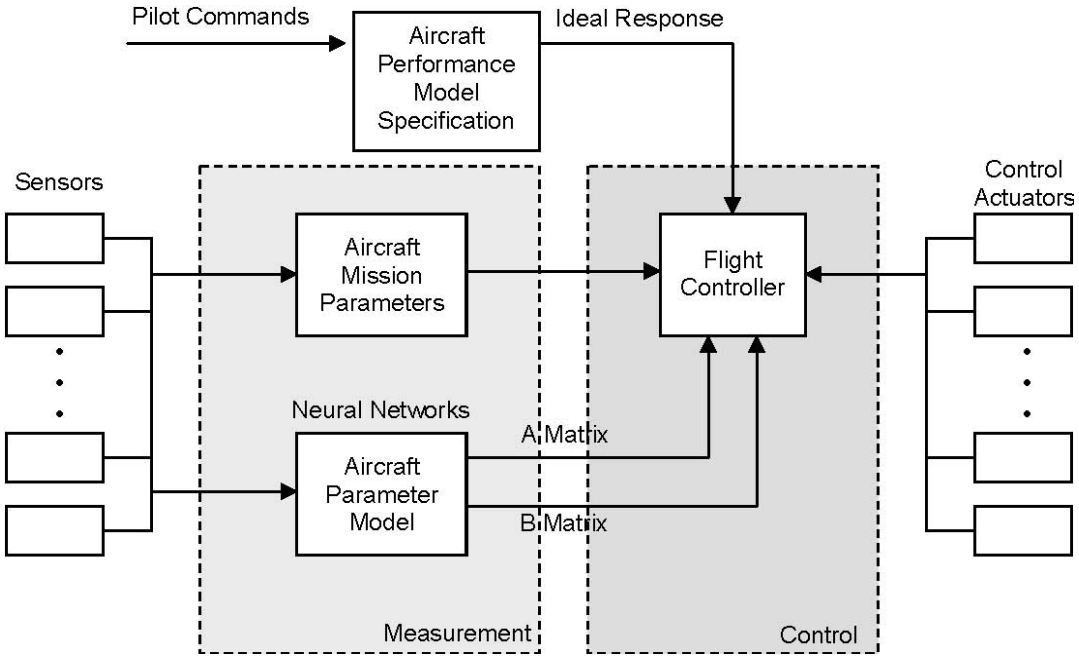


Figure 14 - Application of neural network in IFCS design (after Urnes et al. 2001).

The network design used a pre-trained neural network that may be combined with an additional self-learning neural network. This self-learning network would learn and process the incremental changes to the aircraft plant that may occur under failure or battle damage conditions. The neural network data was provided to an adaptive flight controller that continuously optimizes the control to compensate for damage or failure conditions of the aircraft. The system was implemented on fifteen flights of an F-15 with a test flight envelop with supersonic flight conditions. The system successfully provided continuous monitoring of off-nominal failure or

environment conditions, and immediate assistance to the flight crew and the vehicle control system to regain stable control of the vehicle.

Flight sensor control system

Campa et al. (2002) showed the results of the analysis of a scheme for sensor failure, detection, identification, and accommodation (SFDIA) using experimental flight data of a research aircraft model. The study was based on the use of neural networks (NNs) as online learning nonlinear approximators. The study compared the performances of two different neural architectures. The first one was based on a multi layer perceptrons (MLP) trained with the extended back propagation algorithm (EBPA). The second architecture was based on a radial basis function (RBF) trained with extended-MRAN (EMRAN) algorithms. The scheme had shown to be successful in the detection, isolation, and accommodation of failures 'injected' on a 1/24 scale WVU B777 flight data. The mapping accuracy and the generalization capabilities of both classes of NNs had shown to be critical for the performance of the scheme. The comparison of the two architectures showed that RBF-EMRAN based scheme was slightly better than MLP-EBPA based scheme.

6.3 Application of genetic algorithms

Air traffic planning

Oussedik et al. (2000) presented a new air traffic routes generator based on genetic algorithms. Their objective of developing such route generator was to spread the traffic on new alternative routes due to the traffic growth and congestions in direct and near direct routes. This generator used the information of airspace beacons and sectors. The software generator resulting from the use of the genetic algorithms generated a set of alternative routes that differed from each other in several characteristics, such as geometrical matrices and crossed sectors, with reasonable extra distance compared with the direct route (with the minimum distance). The software generator also produced routes that avoid some congested sectors or restricted areas.

A longitudinal flight controller

Austin and Jacobs (2001) applied genetic algorithms to the design of a longitudinal flight controller for a hypersonic accelerator vehicle that is to be used to launch small satellites. The study examined the capacity of a genetic algorithm in designing a fuzzy logic controller for the task of closed loop flight control. The objective of the design task was to configure the control surface, along with a fixed and preset control structure, through selection of the rule consequents and input scaling. Figure 15 shows the closed loop attitude and trajectory control model for longitudinal flight. The angle-of-attack rule base contained 75 rules in this study.

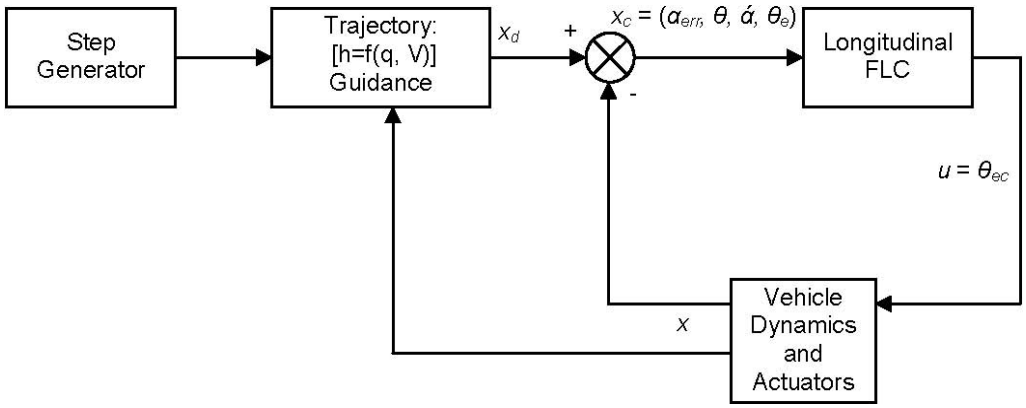


Figure 15 - Closed loop attitude and trajectory control model (after Austin and Jacobs, 2001).

The genetic algorithm uses a collection of simulated flight responses in its formulation of the objective function. This allowed the generation of a controller design without linearization of the vehicle model and dynamics. Stability augmentation was shown through flight simulation at the low-speed end of the hypersonic trajectory and also at a higher flight speed. Emphasis was given on further studies to formulate better guidance rules, minimize computation time, selection of initial conditions and the design objectives.

Optimization of large-scale air combat tactics

Mulgund et al. (1998) in their study developed a software tool for optimizing large-scale air combat tactics using stochastic genetic algorithms. The tool integrated four key components: 1) autonomous blue/red player agents, with their individual aircraft and tactics; 2) an engagement simulator used to play out a tactical scenario; 3) performance metrics reflecting engagement outcome and tactical advantage; and 4) a genetic algorithm (GA) 'engine' for performance based optimization of blue team tactics.

The tool's capabilities were demonstrated throughout the optimization of blue team formation and intercept geometry in a series of tactical engagements. The tactics implementation used a hierarchical concept that built large formation tactics from small conventional fighting units, facilitating the design of tactics compatible with existing air combat principles. In this study, genetic optimization was utilized in four different scenarios. It was found that in each of the scenarios, with respect to casualties, relative advantage, and risk the blue team, that was supported by the genetic algorithm based optimization system, outperformed the red team.

Navigation of the unmanned aerial vehicle

Marin et al. (1999) investigated the use of a genetic algorithm to develop rules that guide an *Unmanned Aerial Vehicle* (UAV) by modeling the amount of uncertainty the UAV faced in terms of probability distributions over grid cells representing terrain. The authors employed the SAMUEL evolutionary learning system to create a set of rules with which to guide the UAV. For training and testing, SAMUEL was provided with terrain data on vegetation, slope, hydrology, roads, and obstacles. The target data consisted of actual tank locations reported every 30 seconds over about an hour. Over thirty tests, the rules developed by the system were able to locate the tank and successfully monitor its location. The authors suggested that further work needed to be done for developing more meaningful measures of effectiveness for the system. The authors would expand the study to include multiple tanks and would attempt to assess the impact of group information on the evolution of rules.

Control of anti-air missiles

Nyongesa et al. (2001) in their study described the application of genetic programming to delay-time algorithms for anti-air missiles equipped with proximity fuzes. The study showed that by applying genetic programming, an evolutionary optimization technique, determination of the timing could be automated and made near-optimal.

Simulation study with two parameter values showed that the evolved algorithms accurately tracked the regions, which in a real missile end-game scenario would correspond to a high probability of destroying the target. Performance measures showed that the root mean square difference between the actual and predicted were less than 0.01% that implied a near optimal prediction. Appendix C summarizes the studies reviewed on utilization of genetic algorithms.

6.4 Hybrid intelligent control systems

Intelligent helicopter flight controller

Zein-Sabatto and Zheng (1997) proposed for an intelligent helicopter flight controller by combining artificial neural network, genetic algorithms, conventional PID controllers, and fuzzy logic algorithms. In this study, the design of the controller was based on experimental data collected from actual helicopter flight. First, a neural network was trained to learn the dynamic characteristics of the helicopter.

Figure 16 below illustrates the block diagram of the neural network based closed loop system. Based on the neural model, the coefficients of a PID controller used for blade angle control were searched by using genetic algorithms. The main rotor speed was designed using fuzzy logic algorithm based on knowledge generated from understanding the aerodynamic theory and analyzing the helicopter experimental data. The intelligent helicopter flight controller was formed by combining the blade angle PID controller and rotor speed fuzzy controller. Figure 17 shows the PID-fuzzy intelligent altitude controller for the helicopter.

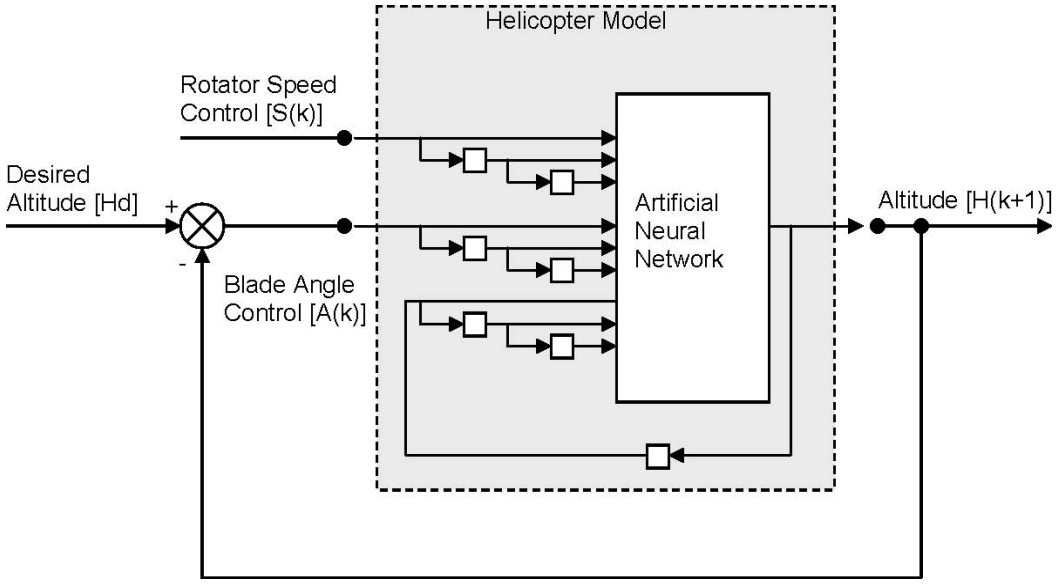


Figure 16 - Block diagram of the NN closed loop system (after Zein-Sabatto and Zheng 1997).

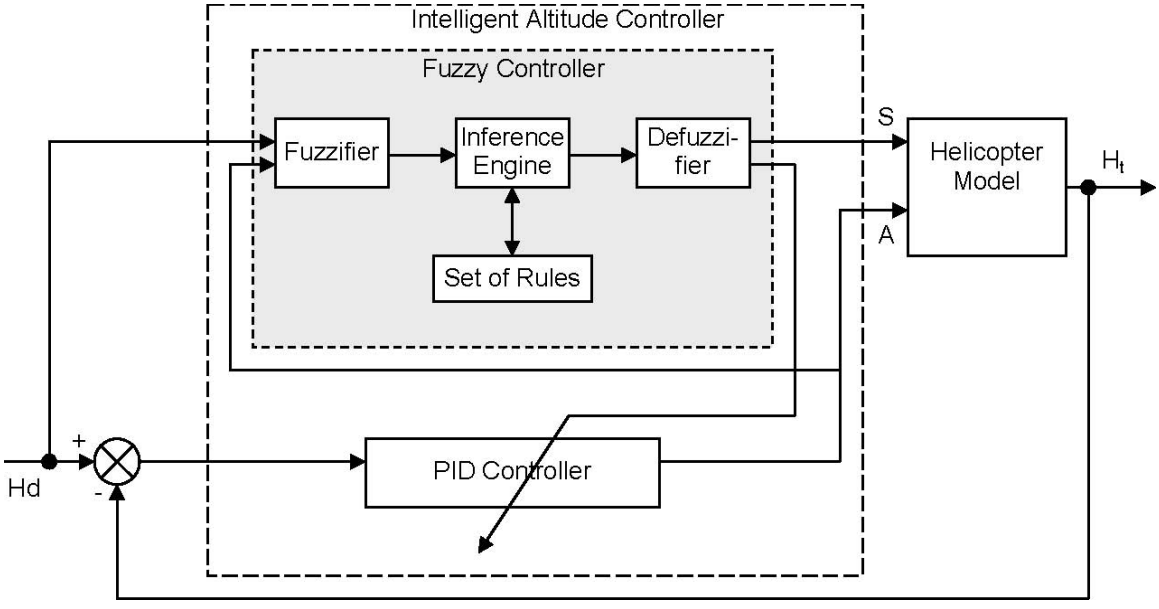


Figure 17 - The PID-fuzzy intelligent altitude controller architecture system (after Zein-Sabatto and

Zheng, 1997).

Simulation results showed that for desired altitude input, the intelligent controller was able to generate proper control signals for both the blade angle and rotor speed controls. The authors stressed on future research by implementing the controller and testing the performance with real flight and then modify and improve the controller.

Fault tolerant flight control system

Idan et al. (2001) introduced an intelligent adaptive neural network based fault tolerant flight control system that blended aerodynamic and propulsion actuation for safe flight operation in the presence of actuator failures. Fault tolerance was obtained by a nonlinear adaptive control strategy based on online learning neural networks and actuator reallocation scheme. Pseudo-control hedging (PCH) was used to address NN adaptation difficulties arising from various actuation anomalies that include actuator position and/or rate saturation, discrete control, actuator dynamics, and partial or complete actuator failures. The control system incorporated a reference model within the control loop. The control system included approximate dynamic inversion and pseudo-control hedging compensation.

A nonlinear single hidden layer NN was used to compensate for the inversion error. The performance of the proposed system was tested on a numerical model of the Boeing 747 aircraft. Simulation of the study showed that by using the adaptive control system the secondary control channels were able to satisfactorily control the speed, pitch rate, and thrust. The adaptive system was also able to successfully identify the model inversion error of the aileron control loop.

Vortex flow control

Joshi and Valasek (1999) proposed for a neural network based controller for bang-bang type vortex flow control nozzles on a generic X-29A. A full state feedback controller was used for the continuous control effecters. The neural network designed was a three layer network with symmetric hidden layers, which optimized a given quadratic performance index. This performance index allowed the designer to specify appropriate weights for states and control effecters to satisfy given specifications. The study also compared the Neural Network Controller to previously designed Model Predictive Variable Structure, and Fuzzy Logic Controllers for the same benchmark problem. Evaluation criteria consisted of closed loop system performance; activity level of the VFC nozzles, ease of controller synthesis; and time required to synthesize controller. The study found that, from a strictly performance point of view, each controller provided good closed-loop performance. The fuzzy based and neural network based controllers

each demonstrated a 9% improvement over the Model Predictive Variable Structure Controller. From an ease of synthesis point of view, the Model Predictive Variable Structure Controller was superior to the Neural Controller and the fuzzy based.

The distinct advantage of the neural controller is seen when the operating conditions depart significantly from the design conditions. The neural controller demonstrated clearly superior robustness characteristics.

Adaptive model-based control of aircraft dynamics

Melin and Castillo (2002) proposed for a hybrid method for adaptive model-based control of nonlinear dynamics systems using neural networks, fuzzy logic and fractal theory. This hybrid system was used for controlling aircraft dynamics systems. For modeling, a generalized Sugeno inference system was used in conjunction with nonlinear differential equations as consequents of the fuzzy rules. Neural networks were used for identification and control while fractal dimensions were fed into fuzzy rule base. Figure 18 illustrates the generic architecture for the adaptive neuro-fuzzy-fractal control.

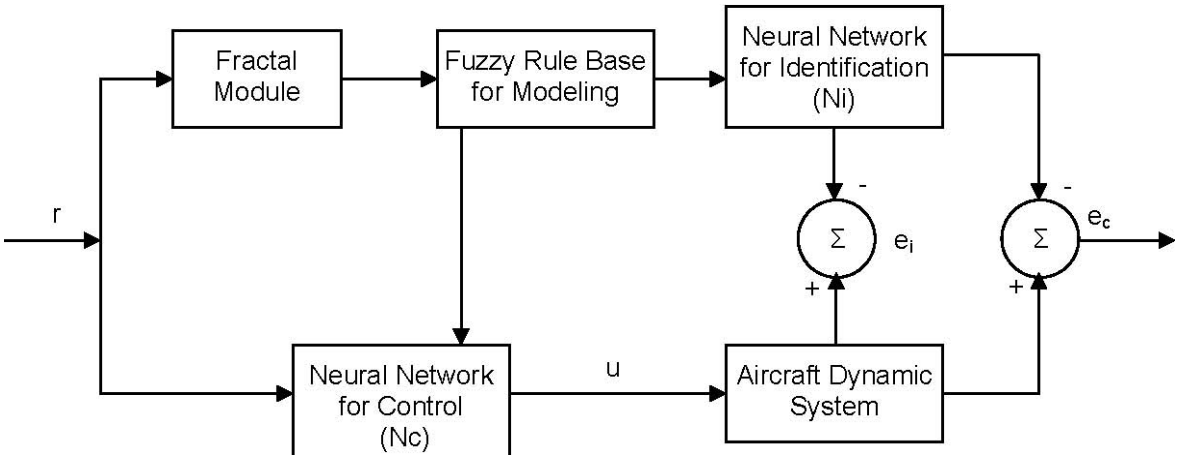


Figure 18 - Generic architecture for the adaptive neuro-fuzzy-fractal controller (after Melin and Castillo, 2002).

The study used three-layer neural networks with Levenberg-Marquardt algorithms. Back propagation technique was used to tune the data. The simulation of this hybrid system showed that identification error was reduced to the order of 10-3 and the final control error using Leveberg-Marquardt was 0.0023.

Positioning of military units

Kewley and Embrechts (1998) developed a fuzzy-genetic decision optimization that solved a problem of positioning military combat units for optimum performance. The optimizer used a simulation model to evaluate solutions, a fuzzy logic module to map simulation outputs to a single fitness value, and a genetic algorithm to search the terrain for a near-optimal combination of unit positions. The results of the study showed that this fuzzy-genetic system outperformed a human expert during a simulated battle. The mean enemy loss was significantly higher when fuzzy-genetic optimizer was used compared the human expert. Further, the mean friendly loss was significantly less for fuzzy-genetic optimization system than for human expert. However, the authors strongly suggested for the optimization system to be used as a decision aide rather than a decision maker.

Target motion analysis

Ganesh (1999) argued that fuzzy logic could offer an enabling technology for automated uncertainty management in the data integration process. In his study, application of this technology to the fuzzy characterization of contact speed with uncertain information was demonstrated, and was shown to provide significant improvement in tracking solution quality for the single-leg target motion analysis problem. The uncertainty in the target end-point location was described by an enhanced area of uncertainty region that was obtained through combination of the derived fuzzy range characterization with conventional probabilistic information.

The author expected that significant benefits would be derived from this technology through (1) increased automation of operator functions, and (2) improved quality of information provided to support informed decision-making; resulting in reduced manning and attendant cost savings.

Complex flight control systems

Wills et al. (2001) proposed for new software infrastructure for complex control systems that exploits new and emerging software technologies. They described a three-level hierarchical control architecture where high-level control incorporates situation awareness, reactive control and model selection; mid-level includes mode transition; and low-level involves stability and control, and augmentation system.

The study also presented an open control platform (OCP) for complex systems, including those that must be reconfigured or customized in real-time for extreme-performance applications.

The OCP consists of multiple layers of application programmer interfaces (API) that increase in abstraction and become more domain specific at the higher layers. A hybrid control strategy was adopted by combining PID and neural networks based controls that operated on flight trajectory (outer loop) and attitude (pitch, roll, and yaw) (inner loop). This OCP was successfully implemented in a helicopter-based test bed.

Rotorcraft control system

Leitner et al. (1998) developed a full authority, six degree of freedom controller of a rotorcraft that provides autonomous, high performance, robust tracking of a specified trajectory. The controller was a combination of traditional PID controller and a neural network based controller. The nominal PID controller was a two time scale input-output-linearizing controller which exploited the well known nonlinearities in the equations of motions, but ignored the variations in the aerodynamically varying quantities. The nominal controller was enhanced with a simple two-layer adaptive neural network that accommodated for the variations in the dynamics and guaranteed ultimate boundedness of the tracking errors in the closed loop. The controller was tested on rotorcraft with highly aggressive, elliptical turn command. The results showed that there were very small tracking errors in both inner and outer loop commanded variables throughout the maneuver. The vehicle remained demonstrably stable throughout the maneuver and all controls remained within their allowable limits. Appendix D summarizes the studies reviewed on utilization of hybrid controllers.

6.5 Classical techniques in adaptive flight controls

The most widely studied approach in nonlinear adaptive flight control involves the use of nonlinear transforms and differential equations that results in system exhibiting linear dynamics (Wahi et al., 2001). This phenomenon is called “feedback linearization”. Feedback linearization theory has found many applications in flight control research.

Meyer and Cicolani (1980) incorporated the concept of a nonlinear transformation in their formal structure to advanced flight control. Menon et al. (1991) introduced a two-time-scale approach to simplify the linear transformations. A special case of feedback linearization control, called ‘dynamic inversion’, has been investigated extensively for application to super maneuverable aircraft (Bugajski et al., 1990; Snell et al., 1992; Buffington et al., 1993).

The above studies showed that dynamic inversion was an effective way of compensating for the nonlinearities associated with high angle of attack flight. However, Brinker and Wise (1996) demonstrated that dynamic inversion technique could be vulnerable to modeling errors. Due to this limitation, a variety of robust nonlinear control schemes were proposed. These

techniques provided robustness to sources of uncertainty that typically include unmodeled dynamics, parametric uncertainty, and uncertain nonlinearities (Brinker and Wise, 1996; Adams and Banda, 1993; Buffington et al., 1993). Krstic et al. (1995) introduced a class of so-called 'backstepping' techniques as an approach to the control of nonlinear systems. Backstepping employed Lyapunov synthesis to recursively determine nonlinear controller for linear or nonlinear systems with a particular cascaded structure. This paradigm afforded the control designer greater freedom in choosing the form of feedback control (Krstic et al., 1994; Kokotovic, 1992).

Parametric adaptive control schemes can be divided into direct and indirect methods. Indirect adaptive control involves online identification of plant parameters. On the basis of this identification, a suitable control law is implemented (Calise and Rysdyk, 1998). In case of direct adaptive control, the parameters defining the controller are updated directly. Studies of Sastry and Isidori (1989) and Kanellakopolous et al. (1991) concentrated specifically on adaptive control of feedback linearization systems.

Dardenne and Ferreres (1998) presented a simple method for the synthesis of robust dynamic feedback of feedforward controllers that satisfy classical time and frequency domain specifications. In Eberhardt and Ward (1999) an indirect adaptive control system approach is demonstrated via the nonlinear six degree of freedom simulation of a tailless fighter aircraft. Huzmezan and Mciejowski (1998), in their study, described reconfigurable flight control of a high incidence research model using predictive control. The paper described a scheme for fault-tolerant control of an aircraft with a high angle of incidence. The study combined the use of high fidelity model of the aircraft with model predictive control, and assumed the availability of information about the faults that had occurred. Looye et al.. (1998) presented the generation of a linear fractional transformation (LFT) based uncertainty model for a civil aircraft that started from a nonlinear dynamic model with explicit parametric dependencies. Boskovic and Mehra (1999) introduced a new parameterization for the modeling of control effector failures in flight control. The approach was illustrated in numerical simulations of the F-18 fighter aircraft carrier landing maneuver. Le Gorrec et al. (1998) demonstrated in improved version of traditional eigenstructure assignment. It produced systems that met robustness requirements. The proposed technique reduced t solving for a quadratic problem under linear constraints.

7. NEURO-FUZZY BASED ADAPTIVE INTERFACE

7.1 Fighter pilot cognition and artificial neural networks

Smith et al. (1991) developed a model that represented the major cognitive states and

decision-making processes of a fighter pilot during the intercept phase of a two-versus-two air combat engagement against a single group of adversary aircraft. In the study, and artificial neural network model was integrated into a hybrid structure containing conventional symbolic logic and algorithmic elements.

A conceptual framework was formulated that defined the situation awareness (SA) construct. The conceptual framework of this pilot engagement consisted of four

Adaptive human-computer interfaces

entities: 1) the environment (e), information (i), knowledge (k), and action (a) vectors. Figure 19 illustrates the flow of data among these entities.

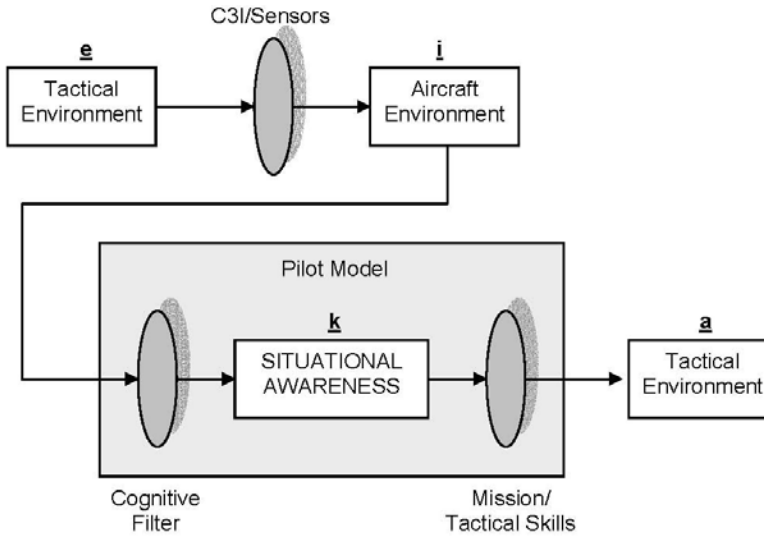


Figure 19 – Situation Awareness Data Flow (after Smith, 1991)

7.2 Cognitive Filter/Mission Tactical Skills

In this study, a database was created that related the time histories of certain pilot cognitive processes that included situation awareness, workload, a decision-making to corresponding traces of tactically relevant environmental variables. Subjective evaluation of 32 trajectories with 288 discrete tactical situations was included in the database. The responses from the database formed the decision vector. A nonlinear algorithmic pilot model was incorporated in

the database. The topology and choice of parameters for the model resulted from a knowledge representation plan based on interviews with air combat tactics and neuro-physiological domain experts. Figure 20 shows the database model.

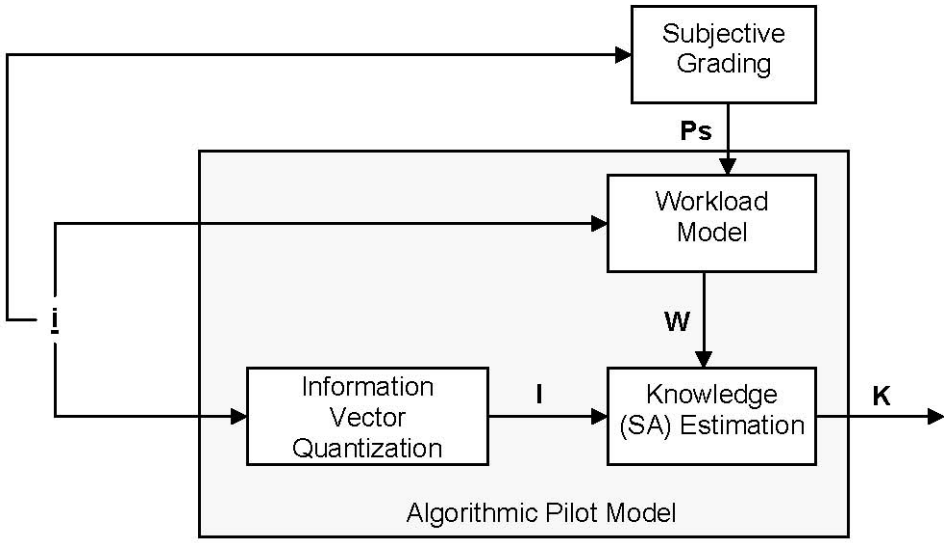


Figure 20 – Database of Pilot Model (after Smith 1991)

The overall simulation of the study consisted of four parts: 1) a threat generation model, 2) a vehicle dynamics model, 3) a sensor model, and 4) the artificial neural network (ANN) model. In the model, the threat generation model provided the capability to present threat aircraft to the ANN model. An unclassified generic fighter aircraft was used as a basis for the fighter dynamics model. A deterministic sensor model provided the link between the threat generation/fighter dynamics and the ANN. The key element of the ANN model was the use of Grossberg's gated dipole. The gated dipole is a biologically motivated structure that is based largely upon the characteristics of the chemical transmitter accumulation and depletion at the synapse. This gated dipole utilizes a tonic arousal level to lead the structure. It also generates an impulse response to the sudden onset and offset of the observed events. Figure 21 depicts the network hierarchy.

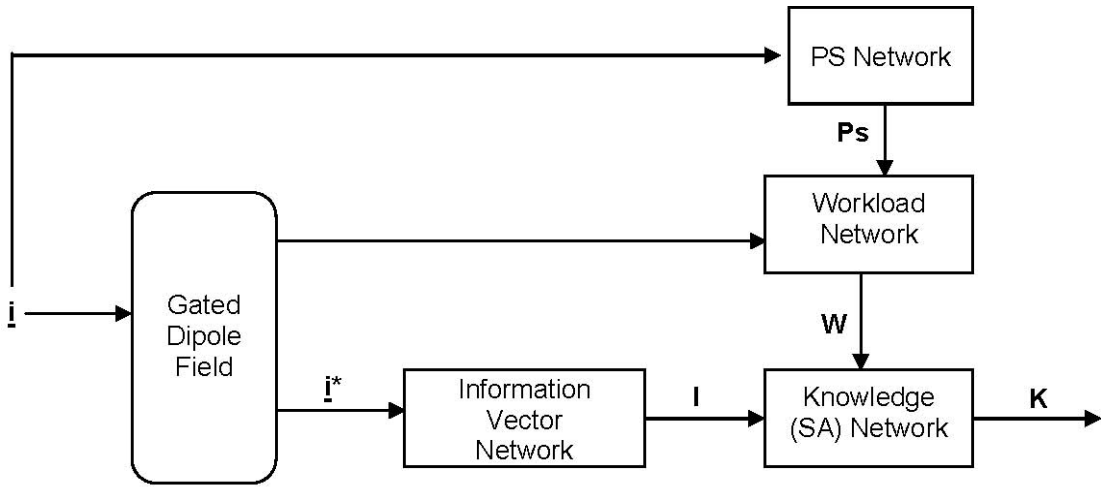


Figure 21 – Network Hierarchy (after Smith 1991)

7.3 Interactive adaptive interface and fuzzy reasoning

Arai et al. (1993) developed an adaptive interface that allowed the interactive adaptation of both the machine and the user. The interface changed the characteristics of the system according to the given task considering the user's skill level, technique, characteristics, and physical condition. The interface is illustrated in Figure 22. The interface was realized according to four kinds of knowledge: 1) knowledge of the system, 2) knowledge of the user, 3) knowledge of the application, and 4) knowledge of interaction between the system and user. Based on this knowledge, the three main elements that were formulated in the model were: 1) user observation system, 2) knowledge database, and 3) adaptive assistance system. Since it is difficult to get the characteristics of the user continuously and adjust from a single observation and the user cannot cope with sudden change of the system, a reciprocal adaptation of both the system and user was proposed. In this case, recursive fuzzy reasoning was used to calculate the assistance level.

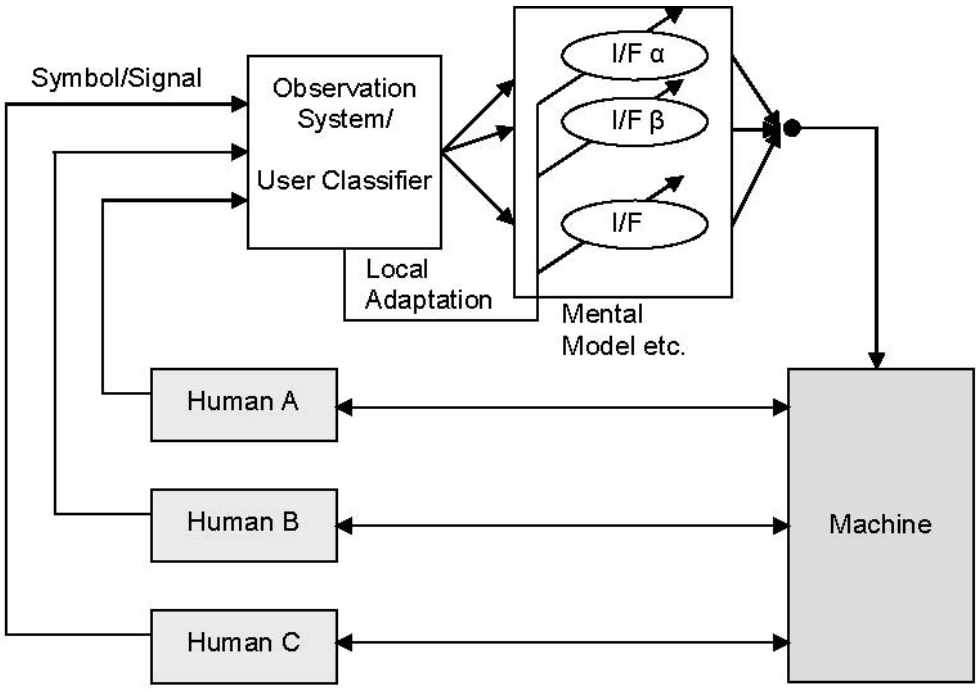


Figure 22 – The concept of a interactive adaptive interface (after Arai, 1993)

Equations (1) and (2) below represent the recursive fuzzy reasoning that was an extension of the simplified fuzzy reasoning.

$$f^{(1)}=\frac{\sum_{i,j}(\mu_{\alpha}^{(1)}\times\mu_{\beta}^{(1)})\,K(\alpha_i^{(1)},\beta_j^{(1)})}{\sum_{i,j}(\mu_{\alpha}^{(1)}\times\mu_{\beta}^{(1)})} \qquad (n=1) \qquad (1)$$

$$f^{(n)}=\frac{\sum_{k=0}^{n}\sum_{i,j}\rho^{k}(\mu_{\alpha}^{(n-k)}\times\mu_{\beta}^{(n-k)})\,K(\alpha_i^{(n-k)},\beta_j^{(n-k)})}{\sum_{k=0}^{n}\sum_{i,j}(\mu_{\alpha}^{(n-k)}\times\mu_{\beta}^{(n-k)})} \qquad (n\geq 2) \qquad (2)$$

Where ρ : Fading factor

The basic assumption was that, by considering the historical changes of the measurement data, it is possible to estimate the user's skill level changes. In the simulation game, the galvanic skin response (GSR) was used as the measurement data. The user's mental stress was estimated from using recursive fuzzy reasoning from the GSR data. From the simulation game it was found that performance under recursive fuzzy reasoning was significantly better than it was in ordinary

fuzzy reasoning.

7.4 Visual perception and fuzzy-neural networks

Hungenahally (1995) implemented a fuzzy neural system in the design of a visual display panel for the purpose of real time operations. This study presents a method of modeling complex information using fuzzy graphs and then integration of the mapped values with higher level learning algorithm for the design of an intelligent warning system. In this proposed system, data acquired from aircraft sensory system were mapped onto fuzzy maps. The information thus represented served as the input to a rule base and/or fuzzy neurons. The fuzzy neural network would process the mapped fuzzy information using fuzzy operators in conjunction with a fuzzy knowledge base. The resulting output of the fuzzy neural network would be displayed in a more formidable way for the human operator or the pilot. The fuzzy neuron comprised of three subunits: 1) the cognizer, 2) the signifier, and 3) the kernel.

The cognizer employs cognitive mapping functions to map the phenomena ‘F’ from a real world domain [xm, xM] to a perceptual domain over [0,1]. The fuzzified inputs were weighted using a function W(k) where ‘k’ was a parameter dependent on the fuzzy knowledge on the cognized data. The shape of the weighing function Wn(k) was determined by the fuzzy knowledge base. The kernel of the fuzzy neuron operated several logical operations on the cognized and weighted information.

This fuzzy neural network model was implemented in a virtual cockpit design or AVID system. The role of AVID was to provide a more ergonomic system for displaying the data and in the development of a complex warning system for the aircraft. Two different systems: 1) with fuzzy neural rule base, and 2) connectionist fuzzy neural network. Figure 23 shows the overall AVID system.

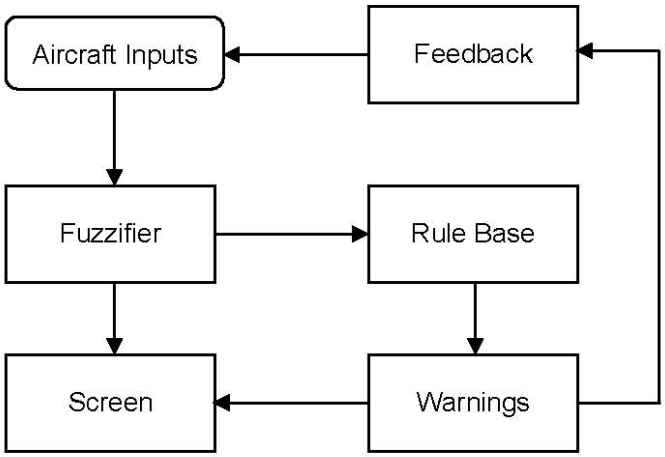


Figure 23 – Overall schematic structure of the AVID system (after Hungenahally, 1995)

The connectionist network had four layers. Layer one was the aircraft input parameters (raw signal data). Layer two fuzzified the data by breaking them into linguistic variables and assigning a mean value. Layer 3 formed the rule base. Each node in layer three served as the rule parameter with inputs from the relevant nodes of layer 2. Layer four nodes would carry the warnings to be stored in priority order and screened. The system was implemented in a aircraft simulator with twenty simulated instrument variables.

7.5 Synthetic vision and fuzzy clustering

Korn and Hecker (2002) studied adverse weather conditions that affect flight safety and efficiency of airport operations. The study focused on the automatic analysis of millimeter wave radar images with regard to the requirements for a sensor based landing. It proposed for a 'electronic co-pilot', which performed the same tasks as the pilot except decision-making. Figure 24 shows the schematic diagram of the electronic copilot model.

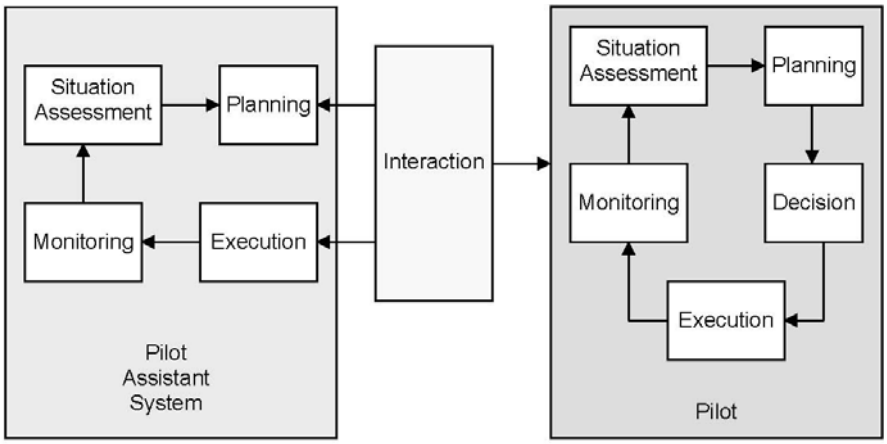


Figure 24 – Electronic co-pilot concept (after Korn and Hecker, 2002)

The key features of such system are situation assessment functions that allow automatic reaction in critical situations. The study focused on the radar image based navigation, i.e., determination of the aircraft's position relative to the runway by analyzing the radar data without using either GPS or precise a priori knowledge about the airport.

8. INTELLIGENT INTERFACES FOR PROCESS CONTROL

8.1 Interactive interface for process monitoring

Arai et al. (1993) designed an interactive adaptation interface monitoring and assisting operator by using recursive fuzzy criterion. Authors defined the concept of interactive adaptation interface as the interface that changing system according the given task considering the user features such as skill level, techniques, characteristics, physical condition. Two kind of interactive adaptation were distinguished: 1) the *Adaptive Assistance Interface*, and 2) the *Adaptive Information Interface* (see Figure 25 for application of the interactive adaptation system). An application of interactive adaptation assistance in the motion level in the adaptive interface for the simulation Air Hockey game was described. In this application, the system changes the automation level according to the user performance and the mental state (stress level).

The level of assistance decreases with the increase in the operator skill level, and increases with the level of the increased stress. Unexpected changes of the interface and assistance level could surprise and confuse user. In order to prevent sadden changes of the interface, the method of assistance level estimation was proposed. This method was based on recursive fuzzy reasoning (Equations 1 and 2) the historical change of the measurement data. The proposed method allows implementing gradual change of assistance level according to the

changes in skill level. The generic structure of the interface architecture consists of three components, the observation system, the knowledge database, and the assistance system.

The observation system monitors the user state. The *Galvanic Skin Reflex G.S.R*. was used to measure the human user's state and to evaluate the stress level. The experimental results showed improvement of the sadden assistance changes problem by recursive fuzzy reasoning.

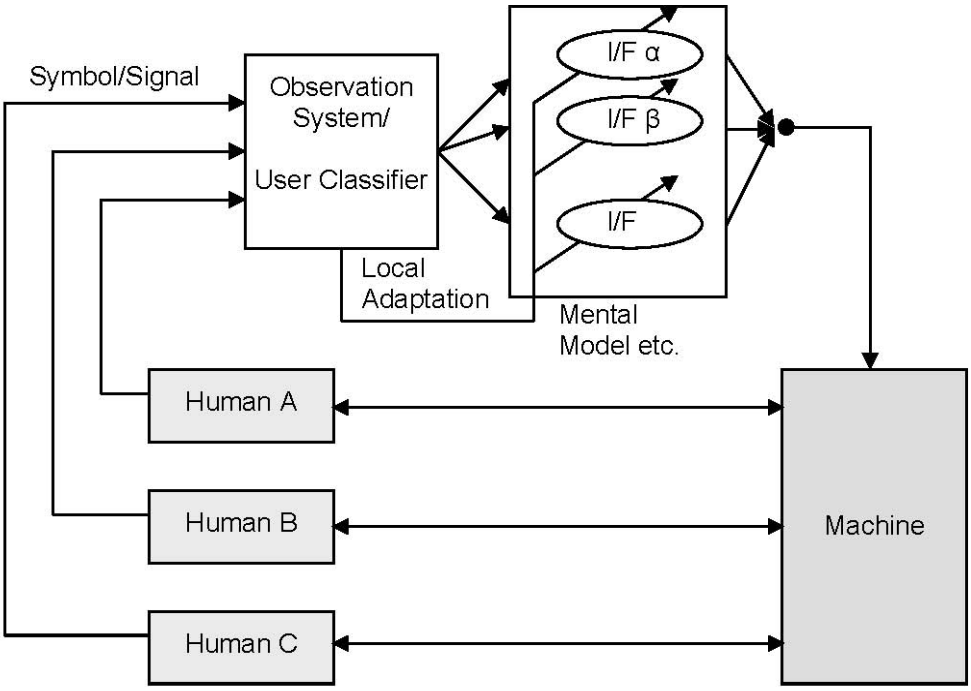


Figure 25 – The interactive adaptive interface (after Arai, 1993)

9. INTELLIGENT INTERFACES: APPLICATIONS

9.1 Decisional Module of Imagery

Kolski et al. (1993) presented the implementation of AI techniques for intelligent interface development in the field of the complex process control. The intelligent interface called the *Decisional Module of Imagery* (DMI) was integrated into an experimental platform and its validation showed that it was technically operational. The "heart" of the DMI is an expert system that manipulates three main objects (the WHAT, WHEN and HOW objects). The interface were developed in the Laboratoire d'Automatique Industrielle et Humaine, Universite de Valenciennes, France. The *Decisional Module of Imagery* (DMI) was integrated into global human – machine

system in the automated process control rooms to obtain an overall assistance tool. The system architecture consist of following main structures: 1) Supervision calculator, 2) Task model, 3) Operator model, 4) DMI, and 5) Expert system.

The *Supervisory Calculator* centralizes all of the process scored data. These data are accessible by both the decision support expert system and the DMI Using these data, the decision support expert system infers information such as predictive, diagnosis or recovery procedures. This set of information is transmitted to the DMI, which selects those that can be presented to the operator. This selection is based on a task model to be performed by the operator, and on an operator's "model" containing information about the operator. The task model was initially restricted to problem-solving tasks and results from a previous analysis of fixed tasks that have to be performed by the operator. This model is based on the general model of Rasmussen, whereby a task is built through four information-processing steps: event detection, situation assessment, decision-making and action. This task model contains a set of process significant variables used by the operator while performing his different tasks. The operator model integrates a set of following ergonomic data: (1) three possible levels of expertise for the human operator (unskilled, experienced, expert), (2) the type of displays associated with each type of operators' cognitive behavior, corresponding to Rasmussen's model, (3) the representation mode associated with each type of display.

The aims of the DMI are as follows: (1) to select the data that can be displayed on the screen, taking into account both the operational process context and the informational needs of the operator; make it possible to operator to supervise the process and to define possible corrective actions; (2) to define the ergonomic parameters associated with the presentation of information for the human operator to understand more easily; and (3) to add the corrective advice to the decision support expert system reasoning and thus to prevent conflicts between the system and the human operator. The expert system consists of an inference engine; a knowledge base on the "What"; a knowledge base on the "When"; a knowledge base on the "How".

9.2 Adaptive information presentation

The DMI adapts itself to the operator by considering information about the following factors:

(1) the various operating contexts of the supervised system,

(2) the operators,

(3) the cognitive and sensormotor tasks of the operators.

The following criteria were established to lead the "What-When-How" decisions of the

interface:

(1) WHAT: All the knowledge and rules needed for each "What-When-How" decision is gathered in knowledge bases that together with inference engine constitute the expert system. The inference engine (Figure 26) handles nine types of fact that represents: 1) What must be displayed, when and how;

(2) The process functioning state, by the use of the facts: "Functioning_ situation", "Situation severity", "Operator's-task";

(3) The type of the operator and his eventual requests: "Operator's _ class", "Operator's _ request"; and

(4) The previous state of the interface: "Previous _ What" (was displayed at the last step).

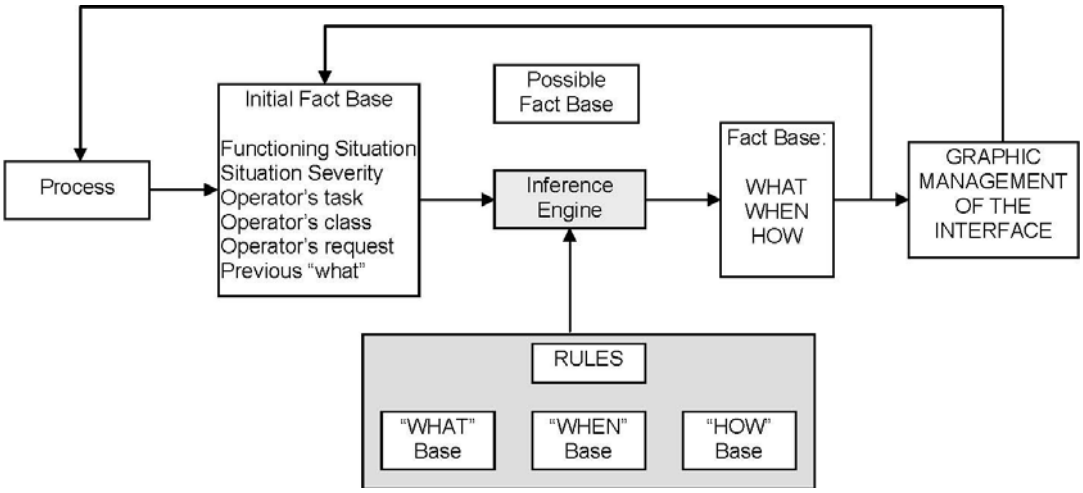


Figure 26 – The Kolski inference engine (after Kolski, 1993)

Facts (2) to (4) are part of the initial fact base. A supervisor provides the expert system with the data necessary for development of this base. The inference engine uses this knowledge base to deduce new facts. The engine starts by inferring on the fact What. The inferred value(s) of the fact What are added to the fact base and then, the facts When and How are deduced. The expert system learns to revise and modify the initial knowledge base by following methodology: 1) The census of all the possible values that are linked to decision criteria about the display is made, icreating the "*Possible Fact Base*"; 2) The connection is built between the registered decision criteria and the potential decisions of the DMI; and 3) Techniques derived from the machine-learning domain are generated to optimize decision trees.

This tool is based on the algorithm ID3 (Iterative Dichotomizer 3) that classifies

decision trees, from the learning set. The experimental platform architecture consists of a set computerized modules, including:

- The process simulator;
- The human operator assistance functionalities, including: 1) a prediction module; 2) an alarm treatment module; 3) an action plan generator; and 4) a justification generator.
- The Decisional Module of Imagery, that integrates: 1) a set of knowledge bases answering the three ergonomic questions: "What", "When" and "How"; and 2) an inference engine that exploits rules contained in the three knowledge bases.
- A graphical task that manage and animate all the views of the interface, using the DMI's answers concerning the "What", "When" and "How" questions;
- A database about the human operators ;
- A supervisor module (to manage the coordination and the communication through the common shared memory).
- A module able to manage failure situations.
- A module able to manage operators' actions and requests.

9.3 Intelligent interfaces for supervisory control

Begg (1994) presented the prototype intelligent graphical user interface developed for application in the real-time supervisory control systems. The main focus of this application to provide the intelligence within interface to assists users in locating, determining, and resolving system problems. The high-level architecture (Figure 27) consists of following: 1) User Interface (UI); 2) User Interface Resources (URI); 3) Graphics Resources (GR); and 4) Intelligence Assistance Resources (IAR). The IGI Channel component represents the central communication channel between operator and *Network Management System* (NMS) and between operator and user interface. *The User Interface Resources* (URI) includes a system model, data logging services, interaction and display techniques, services providing multiple input and output mechanism.

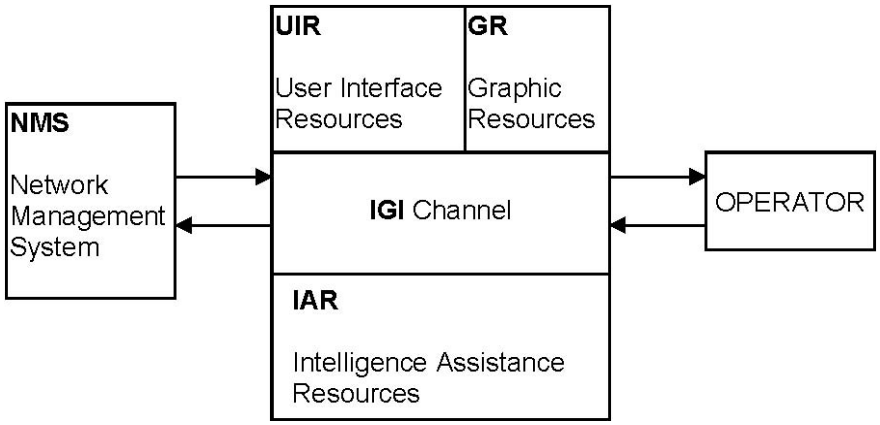


Figure 27 - High-level architecture (after Begg (1994).

The Graphics Resources (GR) includes display and interaction agents. The Intelligence Assistance Resources (IAR) includes a collection of declarative knowledge bases and inference engine that acts on this knowledge. The knowledge bases encapsulates model of the total context in the IGI is operation. This includes models of the domain, user, the user task, and the state of interface. Knowledge for changing the state of the interface comes from human factors guidelines and cases studies results.

Characteristics	**Graphic System**	**Expert System**
Real-time support	System-driven events User-driven events	Inferencing
Process modeling	Task complexity	Reasoning
UI Design	Flexible and configurable good graphics	Interruption of inferencing
Integration	External process interface	External process interface

The implementation requirements listed in the table above were used to determine how the high level architecture. The prototype includes wide range of the graphic techniques for visualization and control of domain information. The variable zoom techniques were used to assist the user in the overcoming “lost in the space” problem. Qualitative overviews comprise abstraction of the low-level data and provide higher level monitoring and problem detection capabilities.

9.4 Intelligent interface for large-scale systems

Yoon and Kim (1996) applied the intelligent interface in aiding the analysis of human

actions and human-machine interaction in the large-scale systems. The system was developed for incident analysis in nuclear power plants in Korea. The intelligent interface was applied in the COSFAH (computerized support system for analyzing human errors). The purpose of this system is to assist the analysts who investigate incidents in large-scale human-machine systems. The support system was developed as a part of computerized HPES (human performance enhancement system) used in nuclear power plants. The architecture of the system (Figure 28) COSFAH was developed to reduce the high mental workload in the composition of an event sequence and to ensure the quality of error analysis. The support system helps the analyst to compose an event sequence.

The interface module provides two major aiding features: the within-record prompting feature and the causal context verification feature. These features are presented via the display and dialog management (DDM) sub-module. There are three inference modules that produce aiding information for event description. The script matching and guidance module provides within-record prompts for composing each line, or record, of the event sequence. The data items composing each record include date, time, record type, error mode for human actions, anomaly indication for system states, and the involved subsystem, part, and its attribute. There is also invisible information associated with each line such as causal relationships with other human actions and system states, related instructions or procedures, and a free-style note for additional description.

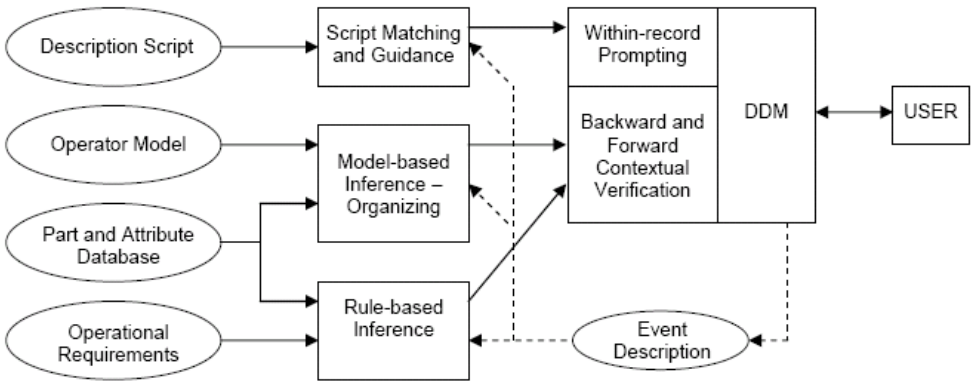


Figure 28 - COSFAH system architecture (after Yoon and Kim (1996).

The data items and their values possess prescribed mutual relationships including requirements or incompatibilities. Due to the relationships and constraints the line of the event sequence is composed. The script matching and guidance module uses a script to assist the user in composing each line.

The system performs two types of causal context verification: 1) backward contextual verification is conducted after each line of event description is put in, and 2) forward contextual verification is started after the first draft of event description is done. In both cases the system examines the consistency and completeness of the event sequence. The system uses the operator model or operational procedures model to check if the activities in the event sequence are logically well composed according to the model. The aid continuously checks the paths through which activates are related to each other against the possible paths allowed in the model. When a mismatch is detected, the aid prompts the analyst to add a record of the missing stage or redefine the relationships between the current record and the previously recorded activities. Two inference modules support the causal context verification feature of both directions: 1) a model-based inference that is based on an operator model, and 2) a rule-based inference that uses operational requirements. Both reasoning modules are supported by a database that contains standardized terms for: system, subsystem, parts, and attributes, and the relationships among them. Operational requirements in the form of production rules are used for the search for missing information in the event description.

9.5 System interfaces that adapt to human mental state

Takahashi et al. (1994) analyzed the effectiveness of a mutually adaptive interface that accommodates the form of human machine interaction according to human mental state.

The adaptive interface was applied to control task difficulty in an example task (X-window-based game, called *X-Jewel*). The architecture of the adaptive interface is presented in Figure 29.

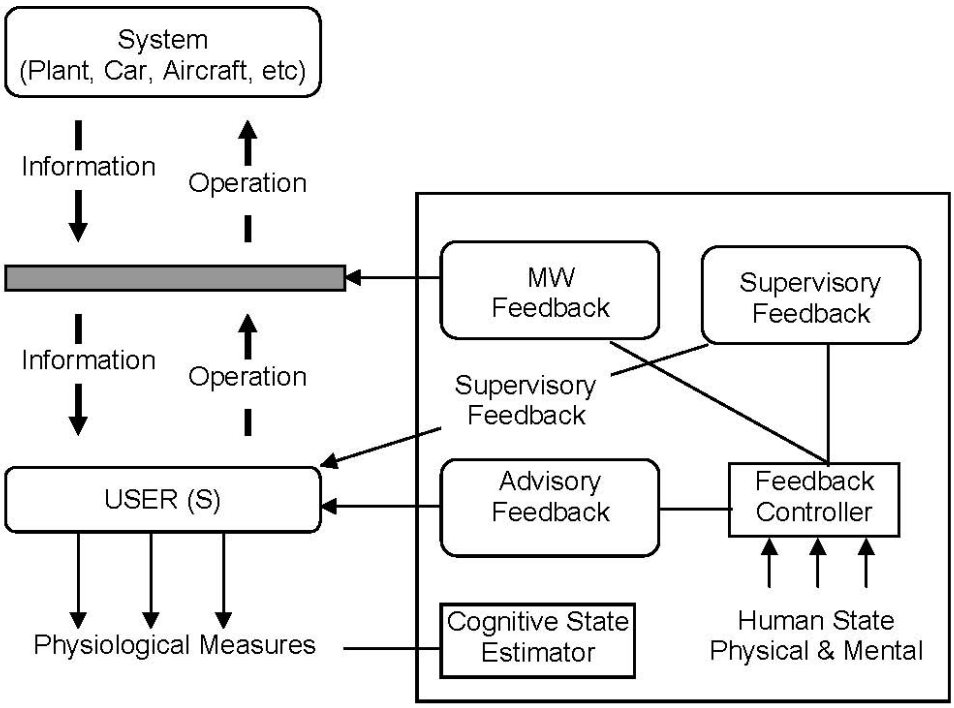


Figure 29 – The architecture of a mutual adaptive interface (after Takahashi, 1994)

The Cognitive State Estimator uses as inputs the physiological measures of the users. The estimated mental workload is utilized by the Feedback Controller to control the form of adaptation. The Mental Work Load (MWL) was used as a representative index of the subject mental state and was estimated by the physiological measures. The physiological measures depicted in the table below were used as estimation of the mental workload. The time margin allowed to complete the task was used as the index representing the MWL. It was assumed that the MWL would increase if the time margin for task completion decreased. The artificial neural network was adopted as the method for empirical modeling the relationship between the MWL and observed physiological measures. The adopted neural network was a three layered feedforward network and is shown in Figure 30. The results of the laboratory experiments showed a significant positive affect on the performance score.

Physiological Features		Classification
Heart Rate (/min)	Absolute Level	1. High 2. Low 3. Normal
	Trend	1. Increase 2. Decrease 3. Steady
Respiration Rate (/min)	Absolute Level	1. High 2. Low 3. Normal
	Trend	1. Increase 2. Decrease 3. Steady
Blood Pressure (mmHg)		1. Increase 2. Decrease 3. Steady
Skin Potential Response (mV)		1. None 2. Low 3. Medium 4. High level
Blink Rate (/min)		1. High 2. Low 3. Normal
Number of Saccado (/min)		1. High 2. Low 3. Normal

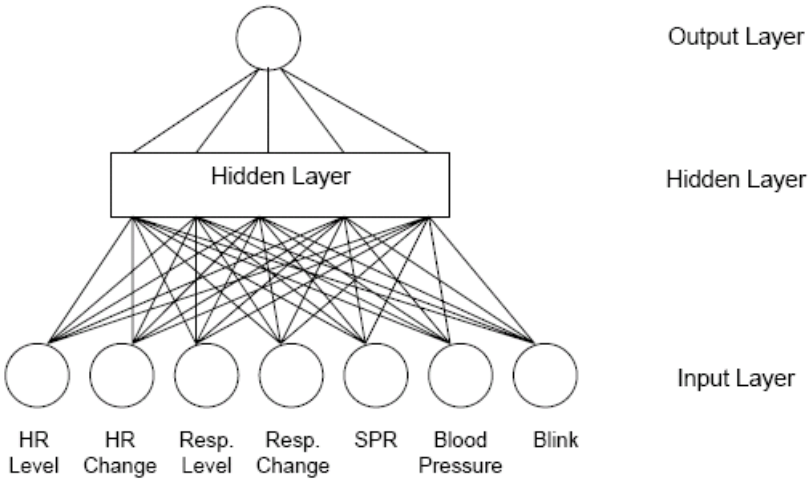


Figure 30 - The configuration of adopted neural network (after Takahashi et al. 1994).

10. ADAPTIVE DECISION MANAGEMENT SYSTEMS

10.1 Adaptive decision support

Fazlollahi et al. (1997) described an adaptive decision support system (ADSS). In an ADSS, the decision maker controls the decision process. However, the system monitors the process to match support to the needs. The proposed architecture evolves from the traditional DSS models and includes an additional intelligent adaptation component. The adaptation component works with the data, model, and interface components to provide adaptive support. The prototype was applied in the forecasting, specifically data analysis and model selection, as

the area of domain knowledge. In this prototype system, the user is provided with the sales data plotted against time and asked to examine the plot and select the most appropriate forecasting model to predict future sales.

The system was built by mapping the conceptual components of the architecture to different files, programs and other features in KnowledgePro software package. KnowledgePro is an environment that supports rapid prototyping in rule-based programming for expert systems. Authors defined ADSS support human decision-making judgments by adapting support to the high-level cognitive needs of the users, task characteristics, and decision contexts. Adaptation was achieved by matching support needs with the system support. The support needs of the user are determined by monitoring the user performance and support history. The support needs of the task and the contexts are identified through monitoring the decision process and selecting the appropriate models. ADSS monitor the decision-making process; diagnose problems/opportunities, and design and implement interventions. Such abilities rest on having knowledge of the specific user, the problem domain, an expert model of the decision process, and strategies for intervention. ADSS provides active participation in the decision-making process. That includes performing tasks such as finding patterns in data, selecting appropriate models, or acting as critiquing agents.

The proposed architecture for ADSS (Figure 31) is an evolution of the Sprague and Carlson model (Fazlollahi et al., 1997). ADSS have three subsystems: 1) user diagnosis, 2) problem-solving, 3) guidance/instruction. Each subsystem incorporates data, model and adaptation component. The user diagnosis subsystem includes information regarding what the user knows and what support the system has already communicated to the user.

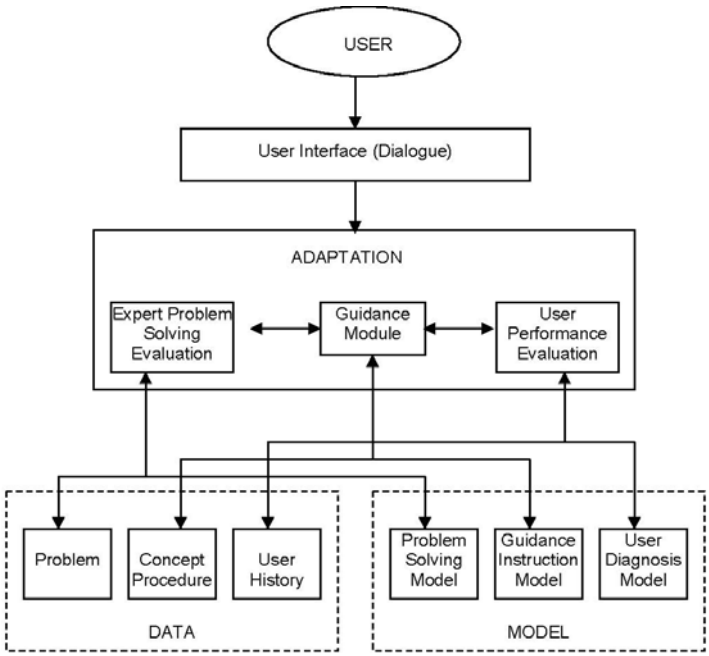


Figure 31 – The ADSS architecture (after Faziollahi, 1997)

The problem-solving subsystem includes the model derived from a theory or stated by the user for appropriately solving the problem. ADSS do not uses the general model of human problem-solving processes to guide their automatic intervention in the decision-making processes. The more attainable descriptive models of specific tasks were used to guide some of the activities of ADSS. The guidance/instruction subsystem includes knowledge about how to intervene in the decision-making processes.

The ADSS architecture addresses the functionalities of ADSS, which are (1) to monitor the decision makers, the decision-making tasks and the decision contexts, (2) to make inferences on the basis of descriptive models, and (3) to intervene at the discretion of the decision maker to provide decision support. Each component of the system divided into subcomponents. Data consists of the problem, the concepts/procedures, and the user history subcomponents. It has data in the form of independent data files and random access memory (temporal data).

- **Data:**
- Problem: The problem data are presented to the user in a graphical format (bitmap) as a time series plot that the user has to analyze:
- Concept/Procedure: the concepts and procedures are assembled in text and graphics formats, in accordance with the problem type and the problem-solving stage requirements.

- User History: this subcomponent deals with temporal data. However, to maintain a cumulative user profile, the data from the random access memory is dumped to a trace (ASCII text/database) file, after every significant event. This file contains data regarding navigation, time stamping, results, performance, etc. In every new session, the trace file from the previous sessions of the user is accessed to adjust for the previously learned concepts and procedures.

- **Model:** The model component consists of rule-based programs (executables), which store the various models used by the system. The model component encapsulates three subcomponents: 1) the problem-solving model, 2) the guidance/instruction model, and 3) the user diagnosis.
  1. The Problem-solving model contains the problem-solving models, represented through associated concepts and associated procedures. This knowledge was modeled by programming in Knowledge Pro's rule-based expert system shell.
  2. The Guidance/instruction model determines the format of the presentation of the concepts and procedures that the user may require. The inference is based on the performance of the user.
  3. The User diagnosis model has rules that diagnose and interpret the user history for determining the strengths and weaknesses of the user in the domain knowledge.

- **Adaptation:** The adaptation is defined through the expert problem/solving evaluation, the user performance evaluation, and guidance subcomponent. All subcomponents are exclusively rule-based, and include the following:
- Expert/Problem Solving: The expert problem/solving evaluation subcomponent associates the problem file name with the problem-solving knowledge rule block. After comparing the problem and the expert's opinion, the subcomponent determines the expert's representation of the required concepts (C E) and the procedures (PE).
- User Performance Evaluation: The user performance evaluation subcomponent examines user history from the trace file and the user diagnosis knowledge. Using the two, this subcomponent determines the concepts (C U) reviewed and procedure (Pu) performed by the user.
- Guidance Module: The guidance subcomponent compares the inferences from the expert problem-solving evaluation subcomponent and the user performance evaluation subcomponent, and generates the deviations for concepts (AC) and procedure (AP).

The system bases its inferences of formats and concepts on the user profile and present user performance (AC and AP). In the prototype system, the outcome for each of the four cases

can be either right or wrong. Therefore, as more information is gathered, the decision tree develops more branches (Figure 32).

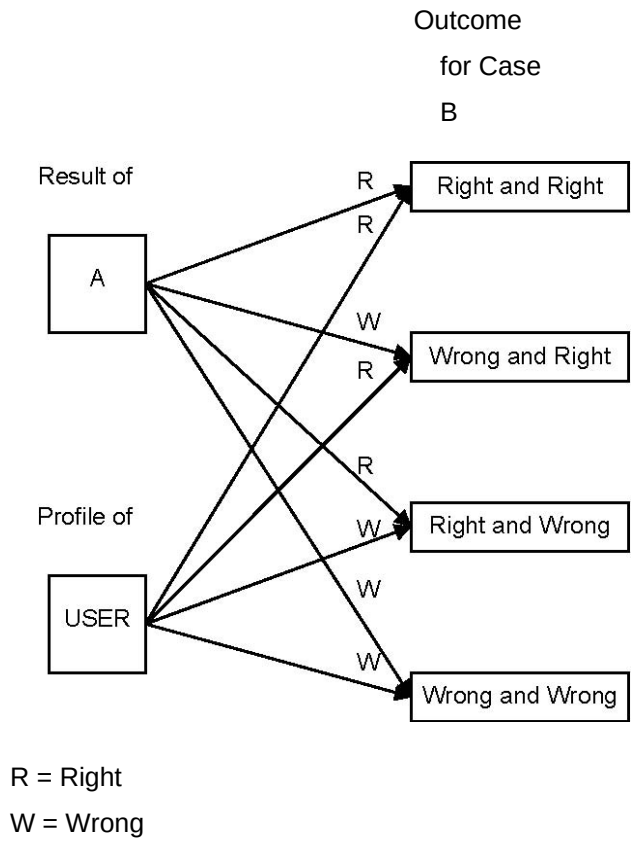


Figure 32 - Example tree (after Fazlollahi et al. 1997)

10.2 Adaptive interfaces based on function allocation

Scallen and Hancock (2001) examined adaptive function allocation in a multitask aviation simulation with tracking, system monitoring, and target identification tasks. In this study three Adaptive Function Allocation (AFA) strategies were examined. In full AFA (auto), the tracking task was completely automated. In one part-task AFA condition, only the vertical component of tracking was automated during AFA episodes while the pilot continued to track horizontally (auto-v). In a second part-task AFA condition, only the horizontal component was automated during AFA episodes while the pilot continued to track vertically (auto-h). During the AFA episode, pilots were cued to the shift in control by an additional display. Monitoring and targeting were

completely manual at all times. The STARFIRE (Strategic Task Adaptation: Ramifications for Interface Relocation Experimentation) adaptive allocation test platform was used. The following tasks were developed to test adaptive function allocation:

- **Task1 – Tracking.** The tracking subtask was located centrally on the HUD. The tracking employs a 3-D pathway-in-the-sky that serves to guide the pilot along a pre-selected route with turns, ascents, and descents in all axes. The pathway is redrawn each second and presents a 10-sec lead. The task goal was to center the aircraft in the path by aligning a nose point symbol with a target symbol that travels through the path. The tracking highway-in-the-sky was imposed on a standard HUD symbology with pitch ladder, altitude, airspeed, and heading indicators. Whereas the tracking pathway-in-the-sky is a 3-D representation, the tracking task itself can be reduced perceptually to two dimensional pursuit tracking.
- **Task 2 – Monitoring.** The system-monitoring subtask is a configuration of five lights (two green lights normally on, two red lights normally off, and one yellow light normally off) and four graduated sliding gauges with criterion-level indicators. The goal for the pilot is to reset the lights or gauges whenever they deviate from normal status by depressing response buttons on the instrument panel.
- **Task 3 - The target identification.** The subtask required the pilot to scan the textured surface for 3-D targets (spheres, cubes, or pyramids). On detecting a target, pilots activated a screen menu, cycle through menu options, and selected the menu item that corresponds to the target shape by depressing switches on the flight stick. Pulling a trigger mounted on the flight stick completed the task.

The results provide support for the implementation of adaptive allocation based on a hybrid model comprising elements of operator performance and mission relevant variables. Implementation of adaptive allocation was an effective countermeasure to the predictable decrease in tracking performance associated with the initial presentation of a surface target.

10.3 Adaptive interfaces based on distributed problem solving

Siebra and Ramalho (1999) developed adaptive interface model based on a distributed problem solving architecture. A *Distributed Artificial Intelligence* architecture consisting of four agents was adopted, the agents being perception, modeling, adaptation, and execution. The Perception Agent receives and processes inputs from the user and the main system to which the interface is attached. The Modeling Agent is responsible for the initialization and updating of the *user model*, which contains information about three generic stereotypes (beginner, intermediate and expert users) plus an individual model for each user. This information is represented by a hybrid formalism combining production rules and objects. The user is characterized by static

(e.g., user login) and dynamic (e.g., user abilities) features and his/her stereotype are dynamically updated by means of production rules.

The *Adaptation Agent* has three basic functions: adapts the interface, fix anomalous actions and sets training sessions to the user. The knowledge necessary to accomplish these tasks is represented in the *domain model*. It contains the interface description (the interface objects, such as windows, icons, buttons, and menus), as well as generic adaptation strategies, including bug library, advising messages, etc. The adaptations are implemented as production rules of the type *IF an error F occurs AND the user level is N THEN execute adaptation A*. The *Execution Agent* implements the execution of actions and presentation of help, advising, error messages and information to the user.

When the user is not able to click in a valid area with a mouse, the possible solutions are: (a) to increase the icon or button size; (b) to consider a valid area around the button or icon, or (c) to propose training session for the user, in the form of a “shot the target” game. Athena was built as a modular, reusable, extensible and portable interface. Due to this, Athena can be easily extended and attached (plugged in) to different systems.

11. GRAPHICAL INTERFACES FOR AVIATION SYSTEMS

11.1 Interface for flight management system

A graphical man machine interface for an Advanced Flight Management System (AFMS) was developed in the Department of Technical Computer Science (LTI) at RWTH Aachen has developed (in close cooperation with NLR (National Aerospace Laboratory, Amsterdam, Netherlands) (Marrenbach, Kraiss, 2000). The new user interface was created to replace today’s Control and Display Units (CDUs). The alphanumerical flight plan editing was replaced by a graphical user interface. A software prototype of such a CDU has been created, using Seeheim model and Statecharts for the definition of this interface.

In the new user interface was used a graphical output device. Furthermore, the system-oriented composition of functions was transferred into an operational structure. Therefore the functionality of the AFMS was partitioned into four levels, called *main task*, *subtask*, *procedure* and *function*. The main control elements of the AFMS are the *main task* and *subtask selection keys, which* are used to enter the main menus and submenus respectively. The *line selection keys* are used to enter the respective procedure and function. The *rotary knob* is used in order to change elements in various selection tapes and the *touch pad* is used in order to control a cursor on the graphical display.

The AFMS provides two ways of access with different functionality: a *function-oriented* and an *object-oriented* access mode. In the *function-oriented mode,* all functions are organized in a so called ’menu tree’. The “menu tree” contains ’branches’ and ’subbranches’ with the

column and line selection keys to access the needed function. The highlighting of the selected menu informs about top level (main menu) menu. The purpose of *object-oriented mode*, i.e. the quick modifications or alterations in flight, allows for direct access to the object on which a function has to be executed by moving the cursor of the touch pad to it. There are only a limited number of functions, which can be used for a selected graphical object. These functions can easily be associated with the CDU´s line select keys. Supplementary to the graphical representation of the flight plan (map-mode, plan-mode, vertical mode) an alphanumerical page is implemented (it is easier for the user to gain an overview of the whole constraint list if it is presented in this way). The benefits of the proposed interface design are as follows:

- The object-oriented approach to design reduces the number of possible functions during selection.
- Fewer keys are necessary, which results in more room for the larger display and in larger buttons, making it less likely to hit the wrong button by mistake.
- The graphical user interface simplifies translating the pilot's idea of a flight plan into the system language.
- The comparison between the Alphanumeric CDU and Graphical ACDU showed that number of actions needed to complete was reduced up to 50%.

11.2 A multi-windows flight management system

Abbott (1997) developed an experimental flight management system (FMS) interface to examine the impact of the primary pilot-FMS interface, the control display unit (CDU), on initial FMS pilot training. The main purpose of the research was the examination of the experimental multi-windows CDU concept based on graphical-user-interface (GUI) techniques. The FMS databases included U.S.-wide information on very-high-frequency omnidirectional ranges (VOR's), low- and high-altitude airway structures, airports, and the geometry of airport instrument landing system (ILS) and runway configurations. Databases also were included for specific standard instrument departures (SID's), standard terminal-arrival routes (STAR's), and approaches for a limited number of selected airports. Performance optimization was based on a Boeing 757 class of airplane that was also the performance model for the airplane simulator used in the evaluation. This optimization provided climb, cruise, and descent schedules; fuel flow estimation; estimated waypoint crossing speeds and altitudes; and waypoint arrival-time estimation. The algorithms also accommodated pilot-entered climb, cruise, or descent speeds; cruise altitudes; and waypoint speed and altitude crossing constraints. The FMS could simultaneously handle four paths or profiles: a primary or active path, a modified active path, a secondary path, and a data-link path. The navigation display (ND) on the simulator instrument panel could display a primary or active path and either a modified active path or a secondary path. Two CDU concepts were developed for this study: a generic, baseline concept and a

graphical-user-interface (GUI) CDU concept. Both CDUs used the same underlying experimental FMS software that included the databases, path-definition routines, and path-optimization techniques. CDU's were physically implemented on a 10-in. diagonal, 16-color liquid-crystal, flat-panel display. The authors indicated that initial design was aimed at evaluating the effects of the multiple windows and direct-manipulation aspects of GUI designs compared to conventional designs. Therefore three major features of GUI were not used in proposed CDU design: pull-down menus, resizable windows, and window scroll bars.

11.3 A navigation hazard information system

Kroft & Wickens (2001) examined effect of three de-cluttering techniques: fixed low lighting, interactive low-lighting and interactive de-cluttering. These de-cluttering techniques were applied to integrated high-clutter digitized displays containing navigation information and air hazard information. Low-lighting displays present one domain of information at a brighter luminance level than the other aspects of the display, while the de-cluttering display removes a domain entirely. Interactive displays allow the user to manipulate, which domain is highlighted, and fixed displays cannot be changed. The fixed low-lighting display did not produce higher accuracy than the baseline large display, nor did it reduce subjects' response times. According to the authors, this lack of a benefit for low-lighting may be the result of a low readability of the low-lighted information, particularly when the ground symbology was low-lighted.

The interactive display produced longer response times that are directly related to the number of time subjects toggled between views. In addition divided attention questions produced longer response times and more toggles than focused attention questions. The authors concluded that the benefit of reduced scanning generally outweighs the cost of increased clutter produced by display integration. This effect (trade off) was more pronounced for divided attention questions than for focused attention questions, as predicted by the proximity compatibility principle.

11.4 Elastic Windows Interface

Kandogan & Shneiderman (1997) described the *Elastic Windows Interface* as an alternative to other windowing systems. The elastic windows design is based on three principles: 1) hierarchical window organization, 2) space filling tiled layout, and 3) multi-window operations. The *hierarchical window organization* supports the user's structuring their work environment according to their roles. It allows users to map their role hierarchy onto the nested rectangle tree structure. Hierarchical grouping of windows is indicated by gradually changing border colors according to the level of the window. This approach was applied in the hierarchical organization of different roles of a university professor: university research and teaching, industry, and

personal. The hierarchical layout clearly indicates the hierarchic relationship between the contents of the windows by the spatial cues in the organization of windows. Hierarchical grouping provides role-based context for information organization. It also supports graphical information hiding capability where window hierarchies can be collapsed into a single icon (or other primitives) making the approach scalable. Collapsed hierarchy of windows can be saved and retrieved, which allows users to reuse a previous window organization

The *multi-window operations on groups of windows* can decrease the cognitive load on users by decreasing the number of window operations. In the case of Elastic Windows, multiple window operations are achieved by applying the operation to groups of windows at any level of the hierarchy. The results of operations are propagated to lower level windows inside that group recursively. In this way, a hierarchy of windows can be packed, resized, or closed with a single operation.

The *space-filling tiled approach* was applied for more efficient use of screen space. In the Elastic Windows, groups of windows stretch like an elastic material as they are resized, and other windows shrink proportionally to make space. Users are given flexibility in the placement of sub-windows in a group. There is no strict horizontal or vertical placement rule within window groups. The extent of window operations is limited to the windows in the same group and their sub-windows. Effects in the upper levels are propagated down to sub-windows recursively.

11.5 Adaptive interfaces in teleoperation

Yoneda et al. (1996) developed an interactive adaptation interface for multimedia tele-operation of a rough terrain crane system. The system has multimodal display that provides force, visual, and acoustic information. The described interactive adaptation interface can adapt the system to an operator considering his/her skill or knowledge, and psychological state. The interface architecture is portrayed visually in Figure 33 and consists of the following elements:

- I/F: I/F transfers operator's command (the angle of joystick) to the goal velocity of each joint.
- Operator Classifier: The evaluation of the operator skill is based on the history of the payload oscillation and joystick command inputs. Operator's psychological state is evaluated by the bionical signal –G.S.R.
- Time constant tuner: Time constant of the operation is regulated by means of the *Recursive Fuzzy Inference*.
    - Multimodal Display: presents information needed for good and easy operation based on the state of payload or the jib.
    - Visual display: a) shadow of the payload; b) arrow – indicates the desirable joystick control direction to suppress the oscillation; c) bars – bars on the right means the current operational angle of the joystick, and the bars on the left means the

desirable operational angle of it; d) side view: state of the jib, wire rope, the payload oscillation, and goal point in the jib hoist.

- Acoustic display: a) oscillation sound: the higher tone- the larger amplitude, and b) job-hoist sound: the higher tone – faster jib motion.
- Force display: operator feels force feedback from the joystick according to the difference between desirable and actual control input. The force display adapt to the operator skill level by changing the strength of the force feed back.

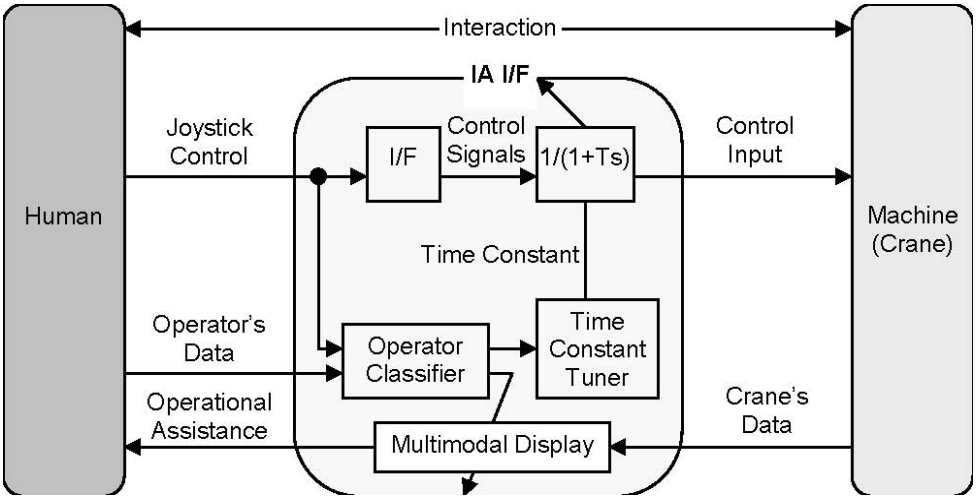


Figure 33 - System architiecture (after Yoneda et al. 1996).

Yoneda et al. (1996) also examined the proposed system on a crane simulator developed for this purpose. The operational experiments confirmed the effectiveness of the proposed crane operational assistance system.

11.6 Adaptive interfaces for driving

Piechulla et al. (2003) proposed an adaptive man–machine interface that filters information presentation according to situational requirements to reduce the driver's information workload. The filter incorporates a projective real-time computational workload estimator which was based on the assessment of traffic situations detected from an on-board geographical database. Workload estimates was refined by data from sensors that monitor the traffic environment and variables of driving dynamics. The prototype was applied to the problem of mobile phone conversations that impairs driving performance. The prototype system was validated in a demonstrator vehicle. The vehicle is equipped with the developer version of a state-of-the-art adaptive cruise control system (ACC), which is based on a radar sensor, and an experimental heading control system (HC) based on computer vision. HC searches for lane markings and employs small forces to the steering wheel, which serve as indicators how to steer

in order to stay in lane.

Workload estimation was made with a software module that predicts the driver's mental strain and reduces additional mental workload resulting from displays, signals, and system messages by postponing less important messages or canceling those messages altogether. The module uses input from car sensors and from an experimental digital map that we call an enhanced database for driver assistance systems (EDDAS). Workload estimation was carried out in a two-stage process. In stage one, a basic estimate was generated by protectively tracking the EDDAS map to identify the oncoming traffic situations and looking up the respective workload indicators for these situations. In stage two, this basic estimate is fine-tuned using information about dynamic aspects of the driving situation. A map-tracking algorithm matches the vehicle position to the EDDAS map and generates a forecast of the route.

Their experiments showed that prototype system is operational in a demonstrator vehicle. Whenever the workload estimate exceeds a threshold value, incoming telephone calls are automatically redirected to the telephone mailbox without notifying the driver. An evaluation field experiment that employed objective and subjective methods of assessing workload yielded promising results in terms of the possibilities of reducing workload by means of the adaptive interface.

12. ADAPTIVE INTERFACES FOR COMPUTER DATABASE APPLICATIONS

12.1 Visual access interfaces

An adaptive interface was used for multi-paradigmatic visual access to databases (Catarci et al., 1996). The aim of this development was integration of different interaction paradigms into a friendly interface for integrated heterogeneous databases. Visual Query Languages (VQLs) based on the visual representations were used to depict the domain of interest and express the related requests. Since certain interaction modalities and visual representations are more suitable for certain user classes, the adaptive interface was applied.

The system proposed by Catarci et al. (1996) suggests to the user the most appropriate interaction modality as well as the visual representation. The interface is adapted according to the user model that provides different visual representations of both data and queries. The visual representations were characterized on the basis of the chosen visual formalisms, namely forms, diagrams, and icons. The system architecture consists of following basic elements: 1) Visual Interface Manager; 2) User Model Manager; and 3) GMDB & Query Manager. The *Visual Interface Manager* provides multiple data representations (form-based, iconic, diagrammatic, and hybrid) and the corresponding interaction modalities together with the possibility of switching among them. For each underlying database a window is available, which can be further subdivided into several child windows, each one displaying the database according to a specific

visual representation. The *Visual Interface Manager* selects the visual representation most appropriate for the user, according to the user model provided by the *User Model Manager*. Such a representation is displayed in the primary child window. The query output, such as text, data or image, appears in a separate child window.

Several visual representations were used in the interface. Form-based representations visualize prototypical forms where queries are formulated by filling appropriate fields. Diagrammatic representations present various types of concepts available in a model with different visual elements. The iconic representation uses sets of icons to denote both the objects of the database and the operations to be performed on them. The hybrid representation is a suitable combination of the above representations. The combination of visual representations is guided by the user needs and the application requirements. Often, diagrams were used to describe the database schema, while icons used either to represent specific prototypical objects or to indicate actions to be performed. Forms were mainly used for displaying the query result.

The *User Model Manager* is responsible for collecting data and maintaining a knowledge-base of the user model components, and provides the most appropriate visual representation and interaction modality according to the user skill and needs. Such a model is dynamically maintained according to the history of the interactions. The database user model consists of three components: 1) the Class Stereotype, 2) the User Signature, and the 3) System Model. The *Class Stereotype* component consists of different classes of database users. User classification was based on following dimensions: professional or non-professional, occasional or frequent, repetitive or extemporary. Those dimensions are used to determine the user features: frequency of the interaction, repetitiveness of the query, structural complexity of the query, and familiarity with the database content. The *User Signature* component contains compressed history of user interactions. The *Class Stereotype* and the *User Signature* components together constitute the individual model of a single user.

The System Model component is the user's own model of the system organization. When the *User Model Manager* construct and store the user's system model, it can suggest the view of the database most appropriate for the user expectations. Moreover, the cost of database search can be reduced during querying. The types of visual representations most appropriate for each user stereotype were defined. As a consequence, once a user has been identified as belonging to a class characterized by a specific stereotype, the system can suggest the visual representation appropriate for that stereotype. However, the user has always the freedom of shifting to a different interaction paradigm based on another visual representation. The appropriateness of a visual representation for a class of users was determined on the basis of the analysis of the advantages and disadvantages of the each interaction paradigms exploiting that visual representation. The system uses two sets of translation algorithms, one for translating a database

expressed in any of the most common data models into a *Graph Model Data Base* (GMDB), and one devoted to implement the consistent switching among different visual representations during query formulation.

12.2 Adaptive interface for generic expert system

Harrington et al. (1996) described an adaptive user interface developed for a generic expert system PESKI (Probabilities, Expert systems, Knowledge, and Inference). PESKI provides a user with knowledge acquisition, verification and validation, data mining, and inference engine tools. PESKI utilizes a Bayesian knowledge base to provide reasoning power that is not designed around a specific application domain. This system allows adaptability to any domain in which it is used. Furthermore, PESKI uses multiple communication modes, allowing a user to select the best possible way to view and input the information contained in the system. The general purpose expert system, such as PESKI, is assumed to have four general types of users: application users, application experts, knowledge engineers, and computer scientists.

The system architecture is composed of three main layers: 1) a Graphical Layer, 2) an Intelligent Assistant Layer, and 3) a System Layer. The *Graphical Layer* creates a graphical work environment for the user. Typical interface objects such as windows, menus, and text entry lines are combined into a functional display that is customized to meet the user's work environment needs. The graphical support is extracted through any number of different interface development tools, and choice of the appropriate tool is based on the architectural platform where PESKI is being used. The Intelligent Assistant manages the interface control tasks through a three-layered construction: the *adaptation layer*, the *adaptive layer*, and the *communications layer*. All transactions between the expert system and the user traverse these three layers for translation and management of data.

The *adaptation layer* acts as an advisor to the user for accomplishing user-performed adaptations. This advisor duty is divided into two tasks: helping the user to make adaptations and advising the user on potential adaptations. When helping the user performs an adaptation, the adaptation layer provides on-line instructions to the user on how to effect the adaptation. The adaptation layer leads the user through the adaptation step by step and gives the user feedback during the process. The *adaptive layer* of the intelligent assistant consists of those elements of the interface that adapt themselves based on perceived user needs, including menus and object layout schemes. The *adaptivity layer* keeps a log of all interface-enacted adaptations for the user's viewing. The *communications layer* of the intelligent assistant is responsible for managing all the data that is passed between the user and the expert system. This layer is equipped with the three communication tools: a natural language interpreter, a graphical interpreter, and a structured text interpreter. With the available communication tools, the user is given the ability to explicitly choose which communication fits their need based on the domain of the data. The

System Layer of the architecture allows connectivity between the user interface and the expert system through a series of tool drivers, which are the link between the interface and the knowledge base, providing the functionality to perform work on the expert system.

12.3 The PUSH Project

Höök et al., (1996) applied the metaphor of a "black box in a glass box" to provide the predictability and transparency for the adaptive interface. In the *Plan and User Sensitive Help* (PUSH) system, the complex inferences of users' goals were hide (in the black box) and a quite simplified view on what is going on (in the glass box) was presented to the user. The user sees a straightforward relation between inferred goal (as unobtrusively presented to the user) and choice of adaptation, therefore basis for predictability are provided. The hypermedia page in this on-line information system provides a complete description of a particular object structured as ordered sequence of typed information entities. Each type of objects in PUSH has its own assortments of information entity types that are used to describe an object of this type. A specific feature of PUSH is that each information entity is a reasonably big portion of hypertext.

To protect users from the information overflow and to help them to find a required piece of information, the system use hiding, i.e. it presents only those types of information entities about the current object that are relevant to the current goal of the user (the goal can be set by the user or deduced by the system). At the same time, to keep the adaptation transparent, the system maintains the stable presentation order of the information entities and never hides non-relevant entities completely: the titles of hidden non-relevant entities are always shown. In the interface, each query generates an answer view that contains all information that possibly may fit the query. This presentation was named as answer page, and represented as a dynamically generated html-page. Some of the information is presented in the view as "hidden" pieces of text or graphics, represented by the mouse-sensitive words or icons. Other types of icons represent navigational maneuvers to other "pages" of information. If the user is not satisfied with the system's decision to show or to hide a particular entity, he can collapse or uncollapse the content of the information entity by clicking on an icon near its title. The information presented to the user is affected in two ways by the selection of a task: 1) the information selection that are deemed relevant for the current task are opened at the time an answer is generated, and 2) follow-up questions are organized into two-level menus, where the first level contains only a few questions, relevant to the current task, and the second level contains all follow up questions.

12.4 Integrated interfaces for web-based applications

Espinoza (1996) proposed an integrated, interactive, multi-modal, World Wide Web (WWW) interface with an adaptive, information filtering system. The combination of multi-modal

interface with adaptive information filtering were applied to solve the problem of information overload and to meet user needs. This interface provided remote access to the PUSH (Plan- and User Sensitive Help) system using web access. The interface was described as multimodal since its produce both graphics and text, and accepts input as text, menu choice, pointing, and selection. Espinoza (1996) created an answer page consisting of graphs and text that the user is allowed to manipulate. The users can navigate in the information space by clicking in the graphs or by posing questions via menus. They can manipulate the answer generated by the system by closing or opening parts of the text. They can also pose follow-up questions on 'hot-words' in the text.

The selection of presented information is based on the user's information-seeking task, which was inferred from their interaction with the system. The user can also actively change the assumed task, and thereby control the adaptive behavior of the system. The realization of the interactive web-based interface and adaptive information filtering is based on a separation of the database and the interface. The database was implemented in SICStus Prolog Objects and serves the remote Netscape clients. The interface was realized using dynamically generated html-pages, and graphs that are generated at the site of the Netscape client using a transferred Java applet.

Interface architecture

The *answer page* is divided into three types of *frames*. Such frames are subparts of the Navigator application window that can be scrolled and resized independently of each other and that each contain a web page: 1) *textual* description of the method consisting of an introduction, a description of the underlying purpose behind the method, and some other information pieces not visible on the screen; 2) *graph* with the process in the middle, its super-process, 'subD', sub-activities, input and output objects; 3) *guide* to the textual description, consisting of the headers to the information pieces. The ones marked as bold are those currently available in the textual description. The system is interactive on several levels. It is interactive at the interface level, allowing the user to manipulate the output from the system. It is also interactive in terms of allowing the user to control the adaptivity:

1) Graphs. The graphics frame serves two purposes. Firstly, it provides a comprehensive view of the information space at the current position; the graphs display all objects related to the current query as well as their relative positions. This gives the user an overview of the domain and also a means for navigation. Secondly, the graphs allow the user to navigate in the information space by clicking on objects in the graph. As the user clicks on an object the view changes to portray the new object and its neighbors, and at the same time the appropriate textual information is retrieved and presented.
2) Text. When the user has clicked on an object, they can also get a textual description of it.

There are many aspects that may be described: how to produce the object, how to work in the process, examples of objects, etc. The user can either ask for a general summary, or just one specific aspect. The users can also manipulate the text frame. They can close or open the information entities through clicking in the guide frame next to the textual frame, and thereby create an answer page that is better fitted to their needs.

3) Hot-words. The user can pose follow-up questions on concepts that are crucial to the understanding of SDP. The term hot-word is used to denote a marked word in the text that is a link to another piece of hypertext. The list of alternative follow-up questions that can be asked about *hot-words* is presented. The hot-words and their associated follow-up questions allow the users to increase their knowledge of SDP. If they are already knowledgeable in SDP, they do not have to read irrelevant information about these basic concepts. 4) Menus. The user can also navigate by composing questions via menus, which is an important alternative to navigation in the graphs. A typical question can be 'describe a process' which would render the answer page. A more specific question could be 'provide an example of process', which would result in an answer page with only one information entity open: the example text.

Several methods were used to create interactive environment for the web: a Java program for client side graphics handling, a CGI program for dynamic generation of web pages, as well as an underlying adaptive database, implemented in SICStus Prolog objects.

System architecture

The system architecture consists of following elements: 1) Netscape Viewer, 2) Page Generator, 3) User Modeling Component, and 4) POP knowledge database. Page Generator sends the query parameters to the POP (the database part of the PUSH system) Prolog program. The user's information seeking task was used, as a tool for determining which information entities would be most relevant to the user in a specific situation. The hierarchy of information seeking tasks was constructed on the basis of a task analysis on user's behavior in their daily work situation. A combination of a user-controlled, and self-adaptive approach to the determination of the current user task was used. A self-adaptive approach is one in which the whole adaptive process is done by the system alone: the system initiates, proposes, decides, and executes the adaptive behavior. The users determine with which task they are initially working. The plan inference (i.e. inferring the users' underlying goal from their actions at the system) is applied continuously to update the user’s task. The user can at any time change the inferred task to some other task.

12.5 Adaptive hypermedia applications

Höök et al. (1997) proposed a service infrastructure for adaptive hypermedia called *edited adaptive hypermedia*. The service involves two types of actors, information brokers and information users, with their respective tasks of collecting, adapting, and reading the information. It was suggested that a solution where individual user interests and preferences are stored in user profiles, available both to the information broker and to the information user. The outgoing information is annotated as to allow for individual adaptations for the information user. Finally, the information user's reading behavior is monitored and feedback is provided to the information broker through the user profile. Höök et al. (1997) proposed to use user models in two ways in the edited adaptive hypermedia service. Firstly, information brokers apply user models to select and filter out relevant information to the reader community, and to structure and annotate the distributed information. Secondly, the information user environment maintains a model of the individual user to provide useful adaptations in the distributed information. These two types of models interact in complex ways. The end user environment only can adapt using such annotations that the information broker has provided. In a closed information domain, an appropriate selection of annotations can be decided upon in advance, but brokers must be provided with feedback on how well the selected annotations worked in practice.

Öquist and Goldstein (2003) described the adaptive RSVP that mimic the reader's cognitive text processing pace by adjusting each text chunk exposure time in respect to the text appearing in the RSVP text presentation window. The *Rapid Serial Visual Presentation* (RSVP) technique is used for dynamic text presentation. RSVP presents the text as chunks of words or characters in rapid succession at a single visual location. This format offers a way of reading texts on a very limited screen space. The exposure time of each text chunk is calculated on basis of the set presentation speed and on how much that can be displayed in the text presentation window. The adaptivity of RSVP is based on the eye-mind hypothesis (Öquist et al., 2003) i.e. that the eye remains fixated on a text chunk as long as it is being processed, the needed exposure time of a text chunk can be assumed proportional to the predicted gaze duration of that text chunk.

Öquist et al. (2003) developed two adaptive algorithms in order to decrease the task load. The first algorithm adapts the exposure time to the content of the text chunks whereas the second also looks to the *context* in the sentences. In content adaptive mode, the exposure time for each text chunk is based on the numbers of characters and words that are being exposed for the moment. In context adaptive mode the exposure time for each text chunk is based on the following: the result of content adaptation, the word frequencies of the words in the chunk and the position of the chunk in sentence being exposed. The width of the RSVP display window was 25 characters wide with the text presented left justified in a 10-pt. sans-serif typeface.

Brusilovsky (1996) described the ELM-ART (ELM Adaptive Remote Tutor) system which

is a web-based Intelligent Tutoring System (ITS) to support learning programming in Lisp. In ELM-ART several adaptive techniques were applied to support students in navigation and learning of course materials. ELM-ART was considered an on-line intelligent textbook with an integrated problem-solving environment. The course material contained in ELM-ART is stored in hypertext form. ELM-ART provides many methods for browsing the course. The system uses two adaptive hypermedia navigation support techniques- adaptive annotation and adaptive sorting of links.

Adaptive annotation uses visual cues (icons, fonts, and colors) to show the type and the educational state of each link. The system maintains an individual permanent model for each registered student. The student's version of ELM-ART uses to distinguish several educational states for each page of material (including problem, example, and manual pages): the content of the page can be known to the student, ready to be learned, or not ready to be learned (the latter case means that some prerequisite knowledge is not yet learned). The icon and the font of each link presented to the student are computed dynamically from the individual student model. They always inform the student about the type and the educational state of the node behind the link. Adaptive sorting is used to present similarity links between cases. Since the system can measure the similarity between each two cases, it can also sort all cases related to the current one according to the similarity values. Links are presented in sorted order - the most relevant first - so the student always knows what the most similar cases are.

The system also provides the *Intelligent Problem Solving Support*. ELM-ART predicts the student way of solving a particular problem and finds the most relevant example from the individual learning history. Answering the help request (e.g. show example), ELM-ART selects the most helpful examples, sorts them according to their relevance, and presents them to the student as an ordered list of hypertext links. The most relevant example is always presented first. Furthermore, the page adaptiveness was applied. All pages presented to the user are generated adaptively "on the fly" when the user requests them. To generate pages, the system uses the text of the course, and knowledge about the structure of the course. When assembling a page of the course, ELM-ART extracts the text of the requested unit from the HTML file and generates the rest of the page (header, footer, hierarchy links and content based links) from the knowledge base. The situation with reference manual pages is even more flexible, because for most of them not only links, but the content itself is generated from the knowledge base. With this approach, all adaptive features of ELM-ART presented above such as additional headers for not-ready-to-be-learned pages or adaptive annotation of links according to their educational state can be easily implemented.

12.6 Auto-adaptive multimedia interfaces

An *Auto-Adaptive Multimedia Interface* (AAMI) was proposed for process control (Viano et al., 2000), by using the general-purpose framework for applications in the *Electrical Network Management* and in the *Thermal Power Plant Supervision.* The described work was carried out in frame of the European ESPRIT project: AMEBICA. AMEBICA is a generic adaptation system that maps events of discrete levels of significance - at the input – to appropriate rendering characteristics at the output. AMEBICA has two interfaces: 1) *the Process Model Agent,* and 2) the *Abstract Rendering Interface* that allow it to interact with its environment. The *Process Model* identifies domain specific occurrences in the operator or system environments and triggers AMEBICA. The *Abstract Rendering Interface* takes general commands from AMEBICA and renders them in domain dependent representations. The interface architecture consists of the following components:

- *Process Model Agent*, which monitors and acts on the process information using its knowledge of the process to translate system dependent calls to AMEBICA calls.
- *Media Agent* that is responsible for rendering of a stream of process information. (contains design time mappings*),*.
- *Rendering Resolution Agent*, which interacts with the *Human Factors Database, Environmental Agent and Operator Agent* to decide upon the best renderings for a certain situation.
- *Environmental Agent* that captures information on the current environmental conditions in the control room.
- *Human Factors Database;* a set of key HCI heuristics used to help decide the most appropriate rendering.
- *Presentation Agent,* which has a continuously updated view of resource usage on the interface in all media.
- *Media Allocator Agent* that makes the final decision on the choice of rendering, based on interactions with Presentation Agent.
- *Operator Agent* that monitors and logs the operator's actions (mouse clicks and keyboard operation).

Adaptation rules are shown in the following matrix.

Operator Response	**Process Status Normal**	**Process State Disturbed High Information Rate**	**Process State Disturbed Low Information Rate**
Normal	OK Process triggering only	OK Process triggering only	OK Process triggering only
Delayed (relative to expected response)	(1) Inattentive Accentuate presentation	(4) Overloaded. Filter information Simplify presentation	(7) “Frozen” Repeat recent information. Try alternative representation
Erratic (occasionally wrong displays or commands)	(2) Inattentive Accentuate presentation (specific)	(5) Overloaded. Simplify displays, Remove information	(8) Partial loss of comprehension Switch modality
Disorganized (constantly wrong display or commands)	(3) Confused, loss of control Go to overview presentation	(6) Sever loss of control “Voice of God”	(9) Complete loss of comprehension. Go one level up. Summarize information

A set of conditions that triggers adaptation was also proposed as a function of the deviations related to the human operator, machine, and the process. For a process, three distinct system states were identified: 1) normal process status, 2) process state disturbed with high information rate, and 3) process state disturbed with low information. The system identifies the operator deviation states on the basis of the inadequate operator responses. The following set of the operator states were differentiated: 1) normal state, 2) delayed (relative to expected responses); 3) erratic (occasionally wrong display or commands), and 4) disorganized (constantly wrong display or commands). The following matrix of adaptation rules were developed on the basis of the sets described in the table above. The following adaptation techniques were developed for the prototype system:

- Highlighting of relevant information according to the actual "scenario". Dynamical detection of the most pertinent and relevant information should manipulate the display parameters to highlight it.
- Adaptation of the display space organization and optimization according to the current "scenario".
- Adaptation of the information representation by selection and manipulation of presentation modalities. Incoming information can be displayed by using a set of alternative modalities selected on the basis of the current "scenario" and of the interface status.
- Time organization. The adaptation action follows the evolution of the "scenario" over time.

12.7 Adaptive interface for knowledge retrieval system

Nguyen et al. (2000) developed an active interface prototype for the *Knowledge Retrieval System*. The prototype under development was planned to test on the *Unified Medical System* (UMLS). The interface agent is based on the *Core Interface Agent* (CIA) architecture. The purpose of this architecture is to provide assistance to the user by maintaining an accurate model of the user's interaction with the target system environment. Interface agents autonomously react to changes in user intent as well as changes to information sources and proactively and dynamically constructs the appropriate queries for the various (heterogeneous) source. The adaptive interface suggest to the user additional related information. A user interacts with a target system (e.g. a medical database querying system), typically via direct manipulation interface such as through menu selections, mouse clicks, and button pressed. The user's interaction with the target system is reported to CIA architecture as observations. The interface agent uses these observations to infer what a user is doing within the environment. Based on the knowledge of the environment that the active user interface has and the user's current interactions with the system, the interface agent determines the user's goal with the highest expected utility and offers a suggestion to the user via the target system.

The user model consists of three components: 1) a user profile; 2) Bayesian network model; 3) utility model. The user profile is used to store the static knowledge about the user, including his demographic data, and skills. The Bayesian network model was applied to capture the uncertain, and causal relationship between the preconditions, goals and actions (the goals were decomposed in multiple actions with many pre- and post-conditions). The Bayesian network model consists of the *action network* and the *ontology network*. The action network was built from the user's natural language queries and relevant feedback on the results given by the system. It was maintained on regular basis by a fading technique. The action network captures the user's

interests, which are controlled in the ontology network. When the action network is faded, the user's actions that persist over time are inferred and copied to the ontology network. The ontology network captures the information regarding the user's long-term interests as well as the information regarding the interrelation among the subjects that the user is looking for.

The utility model was developed to capture the user's utility for having the active user interface performing an action on his behalf to achieve a goal. It was used to support the shifting in user long-term preferences and interests. The utility model contains: 1) the utility function over a set of requirements for active user interface *(UAUI_requirement);* 2) a multi-attribute utility function over a set of metrics that measure the adaptivity, autonomy, collaboration, and robustness of the active user interface,; and 3) the utility function over a set of requirements for capturing the user long-term interests and preferences (U*LONGTERM_requirement*). This function was defined over a set of metrics that measure the scope of knowledge, the rate of changes, the generality and specificity of the user's studying style.

In described implementation of the user model, authors assumed that both of these utility functions needs to look backwards in history to compute the utility function or an agent that extends the user's original queries by taking only the positive or both positive and negative relevant feedback. The decision concerning how the interface agent will offer assistance and if the observation will be included in the user's long-term interest is based on three thresholds. One of thresholds is for offering assistance, second for autonomously performing action on the user's behalf and the third one for the long-term interests. The thresholds definition was based on the expected utility function. When the U*AUI_requirement* falls below the threshold of utility requirement, the active user interface requests "help" from a set of correction adaptation agents. The correction adaptation agent that is most likely to improve the interface agent's requirement utility will get the chance to correct the user model.

12.8 Adaptive interface for medical data management

Dynamic hot-lists were proposed to use in adaptive interface application for data entry of electronic medical records (EMR) in a general practice (Spenceley et al., 1996). The hot lists (split menu) offers selection of frequently used items in a top section of the menu and then gives the remaining items in a bottom section. A split menu is named as intelligent when its top section alters its values to suit factors of the current context. In this application, the hot list creation was based on the task model formed by machine learning from the 3085 records of visits (Adelaide General Practice database). In the database, the patients' problems were coded using from the GP's electronic medical records system.

Conditional probability model were used for anticipation of drugs treatment based on the co-occurrence of drugs with problems. In the training phase, matrix giving the probability for each

drug given each problem was computed from a subset of the database. The matrix constitutes the task model. All drugs were ranked in descending order of probability based on the patient's already-specified diagnoses. The 12 most probable drugs will be shown as the hot list and then list all drugs alphabetically beneath. For situations where one patient has multiple diagnoses it would be ideal to look at the frequency of co-occurrence of particular drugs with specific sets of diagnosis. An approximation mechanism from the fuzzy set theory for such joint conditional probability was used:

$$P(z_1 \vee z_2 \vee \ldots \vee z_n) \equiv max\{ P(z_1),P(z_2),\ldots,P(z_n)\}$$

The definition above was used as max operator for scoring candidates in the hot list. If a patient has problems z_1, z_2, z_3, drugs will be ordered according to their highest probability of occurrence with on of the three problems at the hot list. The 'add' and 'multi' operators were also considered, where each drug where scored by sum or product respectively, of its probabilities with each of the problem present in the record. The 'add' operator rewards associations of drugs with problems with cumulative fashion, while the 'multi' operator penalizes drugs that are weakly associated with any of the problems present. Hit rate index were used to evaluate the model in the simulation. Hit rate is the percentage of the drug from the hot list selected by user. The results of the simulation revealed that: 1) the multi-method is inappropriate for the data, 2) small but consistent advantage of the add method over the 'max method', and 3) the best hit rate- 68%

12.9 Adaptive user interfaces for stock trading

Yoo et al. (2003) developed the *Stock Tracker*, an adaptive user interface that recommends stocks based on an individual's trading profile. The system utilizes this profile to rank stocks, and it revises the profile based on traces of user behavior. Authors emphasized that stock tracking is a temporally sensitive task that requires the continuous monitoring of numeric variables to detect trends or changes over time. This requires identifying features that capture these trends and employing techniques to gather such information for the user. The Stock Tracker is built on the client-server architecture, with information filtering, record keeping, and adaptation performed on the server. The user interface and related computing are done on the client (Figure 34).

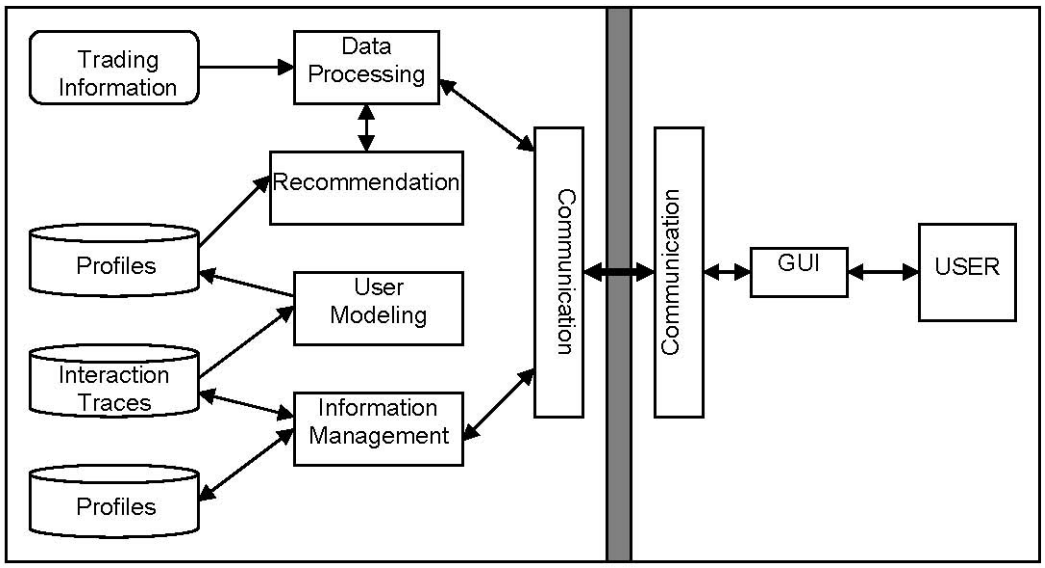


Figure 34 – Architecture of the Adaptive Stick Trader (after Yoo, 2003)

The server contains: 1) the data processing unit, 2) recommendation module, 3) user modeler, 4) information manager, 4) communication unit. The data processing unit converts raw input (i.e., current stock readings and historical trading information) into reports that contain buy and sell recommendations for the user. It relies on the recommendation module to make appropriate suggestions for each stock based on individual user profiles. The user modeler constructs these profiles based on user responses to previous recommendations. The information manager records traces of a user's interactions with the system and also keeps track of user portfolios. The communication unit manages the information into and out of the server.

A client contains: 1) a communication unit, 2) a graphical user interface (GUI) component. The communication unit performs activities that correspond to the server's communication unit. The GUI presents all reports to the user and accepts commands such as buying/selling stocks and viewing portfolios, along with requests for additional financial information on particular companies. The system simulates trading using historical S&P 500 market data; it mimics a real stock-trading scenario by generating information one (simulated) day at a time and letting a trader decide the stocks, if any, to buy and/or sell on each day.

The Stock Tracker includes a graphical interface for presenting stock information, making recommendations, and accepting the user's trading requests. The system's ranked list of recommendations appears in the upper left. Details about the highlighted stock are presented in

the interactive graph in the bottom half of the window. The upper right presents a summary of the current stock, together with the system's recommendation and action buttons for the user to buy or sell the stock. The Stock Tracker's interface was designed to obtain useful feedback through a user's natural interaction. A user can provide positive feedback by purchasing a stock that the system recommends he should buy. By selling the same stock, the trader gives negative feedback. Because more explicit feedback is also helpful, this facility was also provided, but the Stock Tracker can adapt its behavior to users even without such information.

The system bases recommendations on a technical analysis called the Moving Average Convergence Divergence (MACD) that examines the difference between long-term and short-term moving averages to identify crossing points. These points indicate market turns and thus correspond to opportunities for buying or selling stock. MACD was converted into decision rules for recommending different actions: buy, buy warning, sell, sell warning, and do nothing. Each decision rule consists of a set of numeric constraints on temporal stock-trading attributes, such as the rate of increase in the long-term moving average or the difference between the long-term and short-term averages. A decision rule applies when all of its constraints are satisfied—that is, the value of each corresponding attribute satisfies the constraints with the MACD defines the form of the constraints. The Stock Tracker alters its recommendation behavior by incorporating different threshold values.

The Stock Tracker achieves personalized recommendation through the use of individual user profiles that capture trading preferences. A profile consists of four binary classifiers, one for every action other than do nothing, each of which renders a membership decision on each item (i.e., whether it is a positive instance of the class). The system builds classifiers from training examples extracted from traces of the user's interactions. The user can either accept or reject each recommendation. An acceptance indicates that the recommendation was correct and is thus a positive example of the corresponding classifier. Similarly, a rejection produces a negative example. The system also uses positive examples for one classifier as negative examples for others. Although there are many well-known supervised induction algorithms, we opted to devise a new, more efficient algorithm that exploits the fixed structure of the user model (Figure 35).

Learn (*examples*)

For each *classifier* in {buy, buy warning, sell, sell warning}

LearnOne (*classifier, examples, classifier's* constraints)

LearnOne (*classifier, examples, constraints*)

- Sort examples in increasing order according to the attribute value of the next constraint in constraints
- Identify threshold candidates for splitting the examples into positive and negative regions
- Set constraints threshold of classifier to the best split among candidates based on Evaluate (examples, split)
- LearnOne (classifier), Subset (examples, split), Remaining (constraints)

Figure 35 – Example of fixed structure of user model

Each decision rule (classifier) consists of a set of constraints, each of which corresponds to a particular numeric attribute. Every constraint specifies a threshold on the attribute value, above (below) which the constraint is satisfied. The goal of the learning algorithm is to find a set of thresholds that will result in recommendations consistent with the user's actions. The examples were ordered according to increasing attribute values, to evaluate candidate thresholds for a ≥ constraint based on how well they predict positive examples above the threshold and negative examples below it, and similarly for ≤ constraints. Also, since the constraints form a conjunctive set of conditions, thresholds for remaining constraints need only be considered, for examples in the region that satisfies the constraint.

The authors of this article found that the F measure, a weighted combination of precision and recall, provided the best behavior for evaluating candidate splits. Precision indicated the probability that a positive instance. labeled as positive by the classifier. is truly positive, whereas recall gave the probability of correctly identifying all positive instances. In application, precision is the number of positive examples in the correct partition divided by the number of instances in that partition. Recall is the number of positive instances in the correct partition divided by the number of positives in both partitions. The learning is conducted online, that is, after every interaction with the user that yields positive or negative examples, the user modeler updates the user profile. This lets the Stock Tracker adapt quickly to individual traders. To give the system a reasonable starting point, a default model was employed, corresponding to the profile of an average user, which was generated from a default training set that represented feedback from such a user. The weight attached to these default cases allowed varying the degree to which the system relied on them during its model construction.

13. CONCLUSIONS

Contemporary applications of the adaptive human-computer interfaces are spread across a wide rage of different areas, including expert systems, knowledge system databases, World Wide Web pages, tutoring systems, help systems, etc. Currently, the main focus of most of the applications available in the subject literature is on the human user model construction, the control mechanisms, and technical aspects of the interface architecture. However, the cognitive aspects of the user models applied to drive the adaptation are in most cases intuitive and underdeveloped. Furthermore, the knowledge about human information perception and processing perspectives is seldom considered in the design of and information presentation on the available adaptive displays.

In general, one can distinguish four generic elements common to all adaptive interfaces, including: 1) the user model; 2) an adaptation (control) mechanism; 3) a system for assessing the user state, and/or user-interface interaction state, and/or situation/task demands; and 4) the domain model. It should be noted that in some applications the user model consists of two parts: generic or standard user model, and specific (for each user) model (that maintains the specific user features).

Although, there are quite a large number of theoretical papers discussing the above issues in reference to the adaptive interface, these papers are predominantly from the human- or user- oriented areas. At the same time, the literature regarding the design and construction of adaptive interfaces is predominantly from the domain of computer science. These differences are reflected in the variety of names used to designate the adaptive interfaces. An *adaptive interface* implies an adaptation to the user in the human/user oriented areas, while *an intelligent interface* emphasizes the interface intelligence in the context of computer science and engineering.

The information presentation in adaptive interfaces highly depends on the type and area of application. However, the aims of all adaptive techniques applied are obviously the same: to decrease the potential for or eliminate information overload, to present relevant information at relevant moments in time, and finally to assist and improve user performance.

Some common generic factors that influence information presentation in adaptive interfaces can be found. The most important factors that determine the way information is presented are several changing over the time variables such as: user goals, state of user - interface interaction, and task demand and/or situation demands. Two type of the interface adaptability can be noticed: 1) dynamic (active), changing over the time (in some environments very rapidly): adaptation to the effects of interactions between user actions and, task /situation; and 2) passive (inactive): adaptation to the level of the user performance /ability/ knowledge /workload, type of the situation/task.

In case of the passive adaptation the user/task/situation types can be predefined in the interface knowledge base or in models. The dynamic/active adaptation demands analysis of

changing over the time interactions, history of interaction, and information important for these interactions. In some applications with dynamic/active adaptivity the real time assessment of user and/or situation were applied. The following user assessment indices were applied (assessment of the user workload/ability/state): 1) direct physiological assessment: EEG, EKG, GSR, HR, and 2) indirect, situation based assessment of user information needs or workload and user actions/input based.

The above review of literature indicates that aircraft flight and other dynamic systems control design has been to a large extent dominated by the classical system control techniques. While this tradition has produced many highly reliable and effective control systems, recent years have seen a growing interest in applications of robust, nonlinear, adaptive control theory. The use of artificial intelligence computational techniques has dominated in the last decade of control research. This development has been motivated from the desire for enhanced agility and functionality demands that the aircraft or other tactical system perform over an increased range of operating conditions characterized by dramatic variations in dynamic pressure and nonlinear dynamic phenomena. As it has been seen, artificial intelligence techniques have been effectively used either individually or in combination for general aviation, military, or space aircraft attitude, altitude, fault tolerance, angle of attack and other aerodynamic controls. At the same time, these techniques have been successfully implemented in underwater vehicles, weapon systems, and in tactical decision making. However, the review shows that many of the endeavors urge for future research either to overcome limitations in the technology that is currently being used or to develop and implement the technology in the new, challenging real world applications.

14. REFERENCES

Adams, R. J., and Banda, S. S., 1993, An integrated approach to flight control design using dynamic inversion and Mu-synthesis, Proceedings of American Control Conference, 1385-1389.

Amalberti, R., 1992, Reasoning model of fighter pilots, Int J Psychol, 27 (3-4): 604-604.

Amalberti, R., 1991, Decision-making under time-pressure in air combat missions, Med Armees, 19 (6): 359-362.

Amalberti, R. and Deblon, F., 1992, Cognitive modeling of fighter aircraft process-control - a step towards an intelligent on-board assistance system, Int J Man Mach Stud, 36 (5): 639-671.

Amalberti, R. and Menu, J. P., 1985, One approach to optimum flow of information between man and machine, Aviat Space Envir Md, 56 (5): 505-505.

An, P. E., Brown, M., Harris, C. J., Lawrence, A. J., and Moore, C. G., 1994, Associative memory neural networks: Adaptive modelling theory, software implementations and graphical user interface, Engineering Applications of Artificial Intelligence, 7(1), 1-21.

Andes, R. C. and Rouse, W. B., 1992, Specification of adaptive aiding systems, Inform Decis Technol, 18 (3): 195-207.

Arai, F., Fukuda, T., Yamamoto, Y., Naito, T. Matsui, T., 1993, Interactive Adaptation on Interface Monitoring and Assisting Operator Recursive Fuzzy Criterion, Proceedings of IEEE International Workshop on Robot and Human Communication, 448-453.

Austin, K.J. and Jacobs, P.A., 2001, Application of genetic algorithms to hypersonic flight control, Joint 9th IFSA World Congress and 20th NAFIPS International Conference, 4, 2428 -2433, 25-28 July.

Baus, J., Krüger, A., Wahlster W., 2002, A resource-adaptive mobile navigation system, Proceedings of IUI2002: International Conference on Intelligent User Interfaces 2002,ACM Press.

Begg, I.M.; Gnocato, J.; Haman, A., 1994, Architectural Issues in Real-Time Intelligent User Interface Technology, IEEE Network Operations and Management Symposium, 3, 856 -866.

Bennett, K.B., Cress, J., Hettinger, L.J., Stautberg, D., Haas, M.W, 2001, A Theoretical Analysis and Preliminary Investigation of Dynamically Adaptive Interfaces, International Journal of Aviation Psychology, 11(2), 169-195.

Benyon D .R, Innocent P.R., Murray, D.M, 1987, System Adaptivity and the modeling of stereotypes, INTERACT '87, II IFIP Conference on HCI (Elsevier)

Boskovic, J. D. and Mehra, R. K., 1999, Stable multiple model adaptive flight control for accommodation of a large class of control effector failures, American Control Conference, June, 3, 1920-1924.

Brinker, J. S. and Wise, K. A., 1996, Stability and flying qualities robustness of a dynamic inversion control law, AIAA Journal of Guidance, Control, and Dynamics, 19(6), 1270-1277.

Brown, S. M. and Santos, E., Jr., 1999, Active User Interfaces for Building Decision-Theoretic Systems, Proceedings of the 1st Asia-Pacific Conference on Intelligent Agent Technology, 244-253, Hong Kong.

Brusilovsky, P., Schwarz, E., Weber, G.,1996, ELM-ART: An intelligent tutoring system on World Wide Web. In Frasson, C., Gauthier, G., & Lesgold, A. (Ed.), Intelligent Tutoring Systems (Lecture Notes in Computer Science, Vol. 1086). Berlin: Springer Verlag. 261-269.

Buffington, J. M., Adams, R. J., and Banda, S. S., 1993, Robust nonlinear high angle of attack control design for a supermaneuverable vehicle, Proceedings of AIAA Guidance, Navigation, and Control Conference, 690-700.

Buffington, J. M., Sparks, A. G., and Banda, S. S., 1993, Full conventional envelop longitudinal axis flight control with thrust vectoring, Proceedings of American Control Conference, 415-419.

Bugajski, D. J., Enns, D. F., and Elgersma, M. R., 1990, A dynamic inversion based control law with application to the high angle of attack research vehicle, Proceedings of AIAA Guidance, Navigation, and Control Conference, 20-22.

Burdun, I.Y. and Parfentyev, O. M. (1999), Fuzzy situational tree-networks for intelligent flight support, Engineering Applications of Artificial Intelligence, 12, 523-541.

Caldwell, C.W., Martin, T.W., and Cheng, S.I., 1998, Automatic flight path control for nondirectional beacon approach using neural networks, The 1998 IEEE International Joint Conference on Neural Networks and Computational Intelligence, 2 , 980 -985, 4-9 May.

Campa, G., Fravolini, M.L., Napolitano, M., and Seanor, B., 2002, Neural networks-based sensor validation for the flight control system of a B777 research model, Proceedings of the 2002 American Control Conference, 1, 412 -417, 8-10 May.

Catarci, T., Chang, S., -K., Costabile, M. F., M. F., Levialdi, M. F., Santucci, G., 1996, A Graph-based Framework for Multiparadigmatic Visual Access to Databases, IEEE Transactions on Data and Knowledge Engineering, 8 (3), 455-475.

Dardenne, I. and Ferreres, G., 1998, Design of a flight control system for a highly flexible aircraft using convex synthesis, 21st Congress fo the International Council of the Aeronautical Sciences, 89-151, September, Melbourne, Australia.

Eberhardt, R. L. and Ward, D. G., 1999, Indirect adaptive flight control system interactions, International Journal of Robust and Nonlinear Control, 9(14), 1013-1031.

Espinoza, F., Hook, K., 1996, An interactive WWW interface to an adaptive information system. *Paper presented at UM'96* Fazlollahi, B., Parikh, M. A., Verma S., 1997, Adaptive decision support systems, Decision Support Systems, 20(4), 297-315.

Frey, P. R., Rouse, W. B., and Garris, R. D., 1992, Big graphics and little screens - designing graphical displays for maintenance tasks, IEEE Tran Syst Man Cyb, 22 (1): 10-20.

Ganesh, C., 1999, Fuzzy logic-based information processing in submarine combat systems, 18th International Conference of the North American Fuzzy Information Processing Society, 153-157. 10-12 June.

Hancock, P.A., Chignell, M.H., 1988, Mental workload dynamics in adaptive interface design, IEEE Transactions on Systems, Man and Cybernetics, 18/4 , pp. 647-658

Harrington, R. A., Banks, S., and Santos Jr., E., 1996, Development of an intelligent user interface for a generic expert system, In Michael Gasser, ed., Online Proceedings of the Midwest Artificial Intelligence and Cognitive Science Conference. URL: http://www.cs.indiana.edu/event/maics96/Proceedings/harrington.html.

Hettinger, L. J.; Branco, P.; Encarnacao, L. M.; Bonato, P., 2003, Neuroadaptive technologies: applying neuroergonomics to the design of advanced interfaces. Theoretical Issues in Ergonomics Science, vol4, 1/2, pp., 220-238.

Hettinger, L. and Hass, M. (Eds.), 2003, Virtual and Adaptive Environments: Applications, Implications, and Human Performance, New York: Lawrence Erlbaum.

Hilburn, B., Parasuraman, R., & Mouloua, M., 1995, Effects of short-and long-cycle adaptive function allocation on performance of flight-related tasks,In N. Johnston, R. Fuller, & N. McDonald (Eds.), Aviation psychology: Training and selection (pp. 347-353). Hampshire, England: Ashgate.

Hook, K., 2000, Steps to take before intelligent user interfaces become real, Interacting with Computers, 12/4, p., 409-426.

Houlier, S., J-Y, Grau and Valot, C., 2003, A human factors approach to adaptive aids, In: Hettinger, L. and Hass, M. (Eds.), Virtual and Adaptive Environments: Applications, Implications, and Human Performance, New York: Lawrence Erlbaum.

Hungenahally, S.K., 1995, Virtual cockpit: fuzzy neural networks in visual perception Proceedings of IEEE International Conference on Neural Networks, 1, 27-31, 27 November.

Höök, K., Karlgren, J., Wærn, A., Dahlbäck, N., Jansson, C. G., Karlgren, K. and Lemaire, B., 1996, 'A glass box approach to adaptive hypermedia'. *User Models and User Adapted Interaction* 6

Höök, K., Rudström, å. and Waern, A. (1997) Edited Adaptive Hypermedia: Combining Human and Machine Intelligence to Achieve Filter, presented at the workshop: Flexible Hypertext, in Southampton during the Hypertext conference in April 6 - 11th, 1997.

Huzmezan, M. and Maciejowski, J. M., 1998, Reconfigurable flight control of a high incidence research model using predictive control, UKACC International conference on Control, 2(455), 1169-1174.

Idan, M., Johnson, M., Calise, A.J., and Kaneshige, J., 2001, Intelligent aerodynamic/propulsion flight control for flight safety: a nonlinear adaptive approach, Proceedings of the 2001 American Control Conference, 4, 2918 -2923, 25-27 June.

Jeram, G. and Prasad, J.V.R., 2003, Tactile Avoidance Cueing for Pilot Induced Oscillation, Atmospheric Flight Mechanics Meeting, 11-14 August, Austin, Texas.

Joshi, P. and Valasek, J., 1999, Direct Comparison of Neural Network, Fuzzy Logic, and Model Predictive Variable Structure Vortex Flow Controllers," AIAA-99-4279-CP, Proceedings of the AIAA Guidance, Navigation and Control Conference, Portland, OR, 9-11 August.

Kandogan, E. and Shneiderman, B.,1997, Elastic Windows: Evaluation of Multi-Window Operations, Conference on Human Factors in Computing Systems : CHI 97 Electronic Publications

Kanellakopolous, I, Kokotovic, P. V., and Morse, A. S., 1991, Systematic design of adaptive controllers for feedback linearizable systems, IEEE Trans. Automat. Contr., 36(11), 1241-1253.

Keeble R.J. and Macredie R.D., 2000, Assistant agents for the World Wide Web intelligent interface design challenges, *Interacting with Computers-12(4)*, 357-381.

Kewley, R.H., Jr. and Embrechts, M.J., 1998, Fuzzy-genetic decision optimization for positioning of military combat units, IEEE International Conference on Systems, Man, and Cybernetics, 4, 3658 -3663, 11-14 Oct.

Kokotovic, P. V., 1992, The joy of feedback: nonlinear and adaptive, IEEE Control Systems, 12(3), 7-17.

Kolski, C., Le Strugeon, E., Tendjaoui, M., 1993,Implementation of AI techniques for "intelligent" interface development, Engineering Applications of Artificial Intelligence, 6(4), 295-305. Korn, B. and Hecker, P., 2002, Enhanced and synthetic vision: increasing pilot's situation awareness under adverse weather conditions, Proceedings of the 21st Digital Avionics Systems Conference, 2, 11C2-1 – 11C2-10, 27-31 October.

Kroft, P. D. and Wickens, C. D., 2001, Integrating aviation databases: Effects of scanning, clutter, resolution, and interactivity, Proceedings of the 11th International Symposium on Aviation Psychology, Columbus, OH: Dept. of Aerospace Engineering, Applied Mechanics, and Aviation, Ohio State University.

Krstic, M., Kanellakopoulos, I., and Kokotovic, P. V., 1995, Nonliear and adaptive control design, Wiley: New York.

Krstic, M., Sun, J., and Kokotovic, P. V., 1994, Control of feedback linearizable systems with input unmodeled dynamics, Proceedings of the 33rd Conference on Decision and Control, 1633-1638.

Kühme, T., 1993, User-centered approach to adaptive interfaces, Knowledge-Based Systems, 6/4, pp. 239-248.

Langley, P., 1998, User modeling in adaptive interfaces, *Proceedings of the Seventh International Conference on User Modeling,* Banff, Alberta, pp. 357-370.

Le Gorrec, Y., Magni, J., Doell, C., and Chiappa, C., 1998, Modal multimodel control design approach applied to aircraft autopilot design, Journal of Guidance, Control, and Dynamics, 21, 77-83.

Leitner, J., Calise, A., and Prasad, J.V.R., 1998, A full authority helicopter adaptive neuro-controller, Proceedings of IEEE Aerospace Conference, 2, 117-126, 21-28 March.

Looye, G., Varga, A., Moormann, D., Gruebel, G., and Bennani, S., 1998, Robustness analysis applied to autopilot design. I – Mu-analysis of designentries to a robust flight control, Proceedings of 21st ICAS Congress.

Manry, M.T., Cheng-Hsiung, H., and Chandrasekaran, H., Near-optimal flight load synthesis using neural nets, Proceedings of the 1999 IEEE Signal Processing Society Workshop and Neural Networks for Signal Processing IX, 535 -544, 23-25 Aug.

Marin, J.A., Radtke, R., Innis, D., Barr, D.R., and Schultz, A.C., 1999, Using a genetic algorithm to develop rules to guide unmanned aerial vehicles, IEEE International Conference on Systems, Man, and Cybernetics, 1, 1055-1060, 12-15 Oct.

Marrenbach J. and Kraiss, K.-F.,2000, Advanced Flight Management System: A New Design and Evaluation Results. HCI-Aero 2000 - International Conference on Human-Computer Interaction in Aeronautics, September 27-29, 2000, Toulouse, France

Melin, P. and Castillo, O., 2002, Intelligent control of aircraft dynamics systems with a new hybrid neuro-fuzzy-fractal approach, Information Sciences, 142, 161-175.

Menu, J. P., Amalberti, R., and Santucci, G., 1986, The Concept of Intermediate Display, Aviat Space Envir Md, 57 (5): 508-508.

Menu, J. P. and Amalberti, R., 1988, Time pressure (TP) effects information acquisition from head-up (HUD) and head-down (HDD) displays, Aviat Space Envir Md, 59 (5): 483483.

Menon, P. K. A., Catterji, G. B., and Cheng, V. H. L., 1991, A two-time-scale autopilot for high performance aircraft, Proceedings of AIAA Guidance, Navigation, and Control Conference.

Meyer, G. and Cicolani, L., 1980, Application of nonlinear systems inverses to automatic flight control design system concepts and flight evaluations, AGARDograph AG-251 on Theory and Applications of Optimal Control in Aerospace systems, NATO, 10-1 – 10-29.

Millán, J. del R. Mouri no, J., 2003, Asynchronous BCI and Local Neural Classifiers: An Overview of the Adaptive Brain Interface Project, IEEE Trans. on Neural Systems and Rehabilitation Engineering, Special Issue on Brain-Computer Interface Technology, 11(2).

Morris, N. M., Rouse, W. B., and Ward, S. L., 1988, Studies of dynamic task allocation in an aerial search environment, IEEE Tran Syst Man Cyb, 18 (3): 376-389.

Mulgund, S. S., and Zacharias, G. L., 1996, A situation-driven adaptive pilot/vehicle interface, Proceedings of the Third Annual Symposium on Human Interaction with Complex Systems, 193-198.

Mulgund, S., Harper, K., Krishnakumar, K., and Zacharias, G., 1998, Air combat tactics optimization using stochastic genetic algorithms, IEEE International Conference on Systems, Man, and Cybernetics, 4, 3136 -3141, 11-14 Oct.

Mulgund, S., Rinkus, G., Illgen, C., Zacharias, G., and Friskie, J., 1997, OLIPSA: online intelligent processor for situation assessment, the 2nd Annual Symposium and Exhibition on Situational Awareness in the Tactical Air Environment, June 3-4, Patuxent River, MD.

Napolitano, M.R., Cnsanova, J.J., Windon, D.A., II., Seanor, B., and Martinelli, D., 1999, Neural and fuzzy reconstructors for the virtual flight data recorder, IEEE Transactions on Aerospace and Electronic Systems, , 35(1), 61 -71.

Napolitano, M.R., Molinaro, G., Innocenti, M., Seanor, B., and Martinelli, D., 1999, A complete hardware package for a fault tolerant flight control system using online learning neural networks, Proceedings of the 1999 American Control Conference, 4, 2615 -2619 , 2-4 June.

Nguyen, H., Saba, G. M., Santos, E., Jr., and Brown, S. M., 2000, Active User Interface in a Knowledge Discovery and Retrieval System, Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI '2000), Las Vegas, NV.

Nyongesa, H.O., Kent, S., and O'Keefe, R., 2001, Genetic programming for anti-air missile proximity fuze delay-time algorithms, IEEE AES Systems Magazine, 41-45.

Oosterom, M. and Babuska, R., 2000, Virtual sensor for fault detection and isolation in flight control systems - fuzzy modeling approach, Proceedings of the 39th IEEE Conference on Decision and Control, 3, 2645-2650, 12-15 Dec.

Öquist, G., Goldstein, M., 2003, Towards an improved readability on mobile devices: evaluating adaptive rapid serial visual presentation, Interacting with Computers, 15(4),

539-558.

Oussedik, S., Delahaye, D., and Schoenauer, M., 2000, Alternative flight route generator by genetic algorithms, Proceedings of the 2000 Congress on Evolutionary Computation, 2, 896 - 901, 16-19 July, Parasuraman, R., 2002, Adaptive Automation Matched to Human Mental Workload, NATO Advanced Research Workshop on Operator Functional Assessment and Impaired Performance in Complex Work, Il Ciocco, Italy, April 3-7, 2002

Piechulla, W., Mayser, C., Gehrke H., König, W., 2003, Reducing drivers' mental workload by means of an adaptive man–machine interface**,** Transportation Research Part F: Traffic Psychology and Behaviour, In Press, Corrected Proof, Available online 16 September 2003,

Pope, A. T., Bogart, E. H., & Bartolome, D. S., 1995, Biocybernetic system validates index of operator engagement in automated task, Biological Psychology, 40, 187-195.

Prinzel, L.J., Freeman, F. G., Scerbo, M. W., Mikulka, P. J., Pope, A. T., 2000, A Closed-Loop System for Examining Psychobiological Measures for Adaptive Task Allocation*,* International Journal of Aviation Psychology, Vol. 10/4, 393-410.

Rong, J., 2002, Intelligent Executive Guidance Agent for Free Flight, 40th AIAA Aerospace Sciences Meeting & Exhibition, 14-17 January, Reno, NV.

Rothrock, L., R. Koubek, F. Fuchs, M. Haas, and G. Salvendy, (2002) Review and Reappraisal of Adaptive Interfaces: Toward Biologically-Inspired Paradigms, *Theoretical Issues in Ergonomics Science* 3(1), 47-84

Rouse, W.B., Geddes, N.D. and Curry, R.E. (1988). Architecture for intelligent interfaces:Outline of an approach to supporting operators of complex systems, *Human-Computer Interaction, 3(2),* 87-122.

Rouse, W. B., Geddes, N. D., and Hammer, J. M., 1990, Computer-Aided Fighter Pilots, IEEE Spectrum, 27 (3): 38-41.

Rouse, W. B., 1988, Adaptive aiding for human computer control, Human Factors, 30 (4): 431-443.

Saiwaki, N.; Togashi, H.; Tsujimoto, H.; Nishida, S., 1996, An adaptive interface based on physiological indices, Systems, Man, and Cybernetics, IEEE International Conference on, vol.4, 14-17 Oct. 1996, p., 2793 –2798.

Sastry, S. S. and Isidori, A., 1989, Adaptive control of linearizable systems, IEEE Trans. Automat. Contr., 34(11), 1123-1131.

Scallen, S. F., Hankock, P.A., 2001, Implementing Adaptive Function Allocation, International Journal of Aviation Psychology, 11/2, 197-221.

Schram, G., Verschoor, B., Sousa, J.M., and Verbruggen, H.B., 1997, Flight control reconfiguration using multiple fuzzy controllers, Proceedings of the Sixth IEEE International Conference on Fuzzy Systems, 1, 111-117, 1-5 July.

Siebra, S. de A., Ramalho, G.L., 1999, Athena: An User-Centered Adaptive Interface, Proceedings of 8th International Conference on Human-Computer Interaction (HCI'99) Smith, J., Walters, R.V., Goodson, W.L., and Stites, R., 1991, An artificial neural network model of certain aspects of fighter pilot cognition, IEEE International Conference on Systems, Man, and Cybernetics, 3, 1545-1550, 13-16 October.

Snell, S. A., Enns, D. F., and Garrard, W. L., 1992, Nonlinear inversion flight control for a supermaneuverable aircraft, AIAA Journal of Guidance, Control, and Dynamics, 15(4), 976-984.

Spenceley, S.E., Warren, J.R, Mudali S.K, Kirkwood, I.D., 1996, Intelligent Data Entry by Machine Learning of an Anticipative Task Model, Proceedings of the Australian and New Zealand Conference on Intelligent Information Systems.

Soulard P., 1992, Self-adaptive interfaces and multi-modal human interaction for combat systems, Undersea Defense Technology, London, http:// www.artis-acta.com/ARTIS/92UDT/92UDT.HTML

Takahashi, M.; Tsuyoshi, A.; Kuba, O.; Yoshikawa, H., 1994, Experimental study toward mutual adaptive interface, Robot and Human Communication, RO-MAN '94 Nagoya, Proceedings, 3rd IEEE International Workshop on, 18-20 July 1994, p, 271 -276

Tannen, R. S., Nelson, W. T., Bolia, R. S., Haas, M. W., Warm, J. S., Hettinger, L. J., Dember, W. N., Stoffregen, T. A., 2000, Adaptive integration of helmet-coupled multisensory displays for target localization, Proceedings of the IEA 2000/HFES 2000 Congress, 3*,* 77–80.

Terrence S. Abbott, 1997, A Comparison of Two Control Display Unit Concepts on Flight Management System Training, NASA TM-4744, January 1997, pp. 9

Tolle, H. and Ersfi, E., 1992, Neurocontrol: learning control systems inspired by neuronal architectures and human problem solving, Lecture Notes in Control and Information Sciences 172. Springer: New York.

Urnes, Sr., J., Davidson, R., and Jacobson, S., 2001, A damage adaptive flight control system using neural network technology, Proceedings of the 2001 American Control Conference, 4, 2907 -2912, 25-27 June.

Vachtsevanos, G. Kim, W., Al-Hasan, S., Rufus, F., Simon, M., Shrage, D., and Prasad, J.V.R., 1997, Autonomous vehicles: from flight control to mission planning using fuzzy logic techniques, 13th International Conference on Digital Signal Processing Proceedings, 2, 977-981, 2-4 July.

Viano, G., Parodi, A., Alty, J. L., Khail, C., Biglino, I. A. D., Crampes, M., Vaudry, C., Daurensan, V., Lachaud, P., 2000, Adaptive User Interface for Process Control based on Multi Agent Approach. Advanced Visual Interfaces, 201-204.

Vico, F. J., Mir, P., Veredas, F.J., de La Torre, J., 2001, Animal-like adaptive behavior, Artificial Intelligence in Engineering, 15/1, p. 5-12.

Wahi, P., Raina, R., and Chowdhury, F.N., 2001, A survey of recent work in adaptive flight control, Proceedings of the 33rd Southeastern Symposium on System Theory, 7-11, 18-20 March.

Warwick, K., 1996, Intelligent adaptive control, in M. M. Gupta and N. K. Sinha (Eds), Intelligent Control Systems, IEEE Press: New York, 63-85.

Wickens, C.D. (1992). Engineering Psychology and Human Performance (2nd ed.). New York: Harper Collins.

Wills, L., Kannan, S., Sander, S., Guler, M., Heck, B., Prasad, J.V.R., Schrage, D., and Vachtsevanos, G., 2001, An open platform for reconfigurable control, IEEE Control Systems Magazine, 21(3), 49 -64.

Wu, S. –F., Engelen, C.J.H., Babuska, R., Chu, Q. –P., and Mulder, J.A., 2003, Fuzzy logic based full-envelop autonomous flight control for an atmospheric re-entry spacecraft, Control Engineering Practice, 11, 11-25.

Yan, L, Sundararajan, N., and Saratchandran, P., 1999, Fault tolerant flight controller using minimal resource allocating neural networks (MRAN), Proceedings of the 1999 American Control Conference, 4, 2605 -2609, 2-4 June.

Yoneda, M., Arai, F., Fukuda, T., Miyata, K., Naito, T., 1996, Multimedia tele-operation of crane system supported by interactive adaptation interface, Robot and Human Communication, 5th IEEE International Workshop on, 11-14 Nov. 1996, p. 135 –140

Yoo, J., Gervasio, M., Langley, P., 2003, An Adaptive Stock Tracker Personalized trading advise, Proceedings of the International Conference on Intelligent User Interfaces, Miami, Florida, pp. 197-203.

Yoon, W. C., Kim, Y.S., 1996, Aiding the analysis of human actions in large-scale systems: an intelligent interface approach**,** Computers & Industrial Engineering, Volume 30, Issue 3, July 1996, Pages 569-577

Zames, G., 1998, Adaptive control: towards a complexity-based general theory, Autometica, 34(10), 1161-1167.

Zein-Sabatto, S.; Yixiong Z., 1997, Intelligent flight controllers for helicopter control, International Conference on Neural Networks, 2, 617 -621, 9-12 June.

Zhou, Z. Lin, C.-F., and Burken, J., 1997, Fuzzy logic based flight control system for hypersonic transporter, Proceedings of the 36th IEEE Conference on Decision and Control, 3, 2730-2735, 10-12 Dec.

APPENDIX A.

Reference	Type of control	Application	Input	Output	Architecture
Zhou et al., 1997	Flight control	Hypersonic transporter	1. Angle of attack 2. Pitch rate	Stability in hypersonic region Robustness across flight envelop	Fourteen fuzzy inference rules Max-min composition algorithm
Schram et al., 1997	Failure-tolerant control	Civil aircraft simulation	Measurements: 1. Sensor failure 2. Actuator failure	Flight path: 1. Lateral deviation 2. Roll angle 3. Sideslip	Multiple Fuzzy Controllers: 1. Dynamic filter and scaling 2. Fuzzification 3. Inference mechanism 4. Defuzzification
Vachtsevanos et al., 1997	Flight control and mission planning	Autonomous UAV	Route Planner: 1. Distance 2. Hazard 3. Maneuverability Navigation: 1. Waypoints 2. Energy management Fault-tolerance: 1. Sensor failure 2. Component failure	Longitudinal Lateral Vertical Yaw Pitch velocity and control	1. Mission Planner: - Supervisory controller - Route planner - Fault tolerance - Fuzzy navigator 2. Flight controller
Oosterom and Babuska, 2000	Implementation of virtual sensor for normal acceleration	Small commercial aircraft	Dissimilar consolidated sensor readings: 1. Longitudinal motion 2. Sensor noise 3. Atmospheric turbulence	Normal acceleration	Data Generation -Training data - Validation data 2. Fuzzy clustering identification -Takagi-Sugeno model
Napolitano et al., 1999	Development of virtual flight data recorder	Commercial aircraft	Pitch, bank, and heading angles, altitude, airspeed, accelerations	Control surface deflections	1. Neural network simulator: -Multilayer - Back propagation learning algorithm 2. Fuzzy Logic - Fuzzification - Inference - Composition - Defuzzification
Burdun and Parfentyev, 1999	Intelligent flight support	Intelligent pilot-vehicle interface, automatic flight-envelope protection, autonomous (robotic) flight including multiple vehicle systems, and resolution of	Pilot errors, nonstandard flight profile or maneuvers, mechanical failures, wind and turbulence, weather and extreme atmospheric conditions, electromagnetic discharges, extreme		1. Flight event 2. Elementary situation 3. Flight situation scenario 4. Fuzzy situation tree-network

		conflicts in close free-flight air space (Potential)	runway conditions		- Linguistic flight variables - Fuzzy measurement scales - Fuzzy situations - Fuzzy Transitions - Fuzzy branches
Jeram and Prasad, 2003	Active control system	Rotorcraft	1. Dominant frequency, 2. Phase lag, and 3. Actuator rate limit 4. Friction, 5. Radius of motion 6. Bobweight dynamics	Pilot induced oscillation	1. Cockpit control 2. Aircraft state 3. Actuator position 4. Main frequency selection 5. Fuzzy Inference system 6. PIO estimation
Rong, 2002	Optimal and conflict-free flight path guidance		1. Weather 2. Traffic	Flight path	1. Executive agent - Rule-based arbitrator - Traffic conflict evaluator 2. Weather agent 3. Traffic agent
Wu et al., 2003	Intelligent and autonomous flight control system	Re-entry space vehicle	1. Pitch 2. Roll 3. Yaw	Flight trajectory: 1. Angle of attack 2. Sideslip angle 3. Bank angle	1. Fuzzy controller 2. Aerodynamic inversion 3. X38 simulator

APPENDIX B.

References	Type of control	Application	Input	Output	Architecture
Caldwell et al., 1998	landing approach navigation aid	Commercial aircraft	1. Autopilot: - Distance - Bearing - Heading – Airspeed - Altitude 2. Approach: - Inbound course - Turn course - Distance 3. Wind Settings: - Velocity - Direction	1. Drift heading 2. Relative bearing	Feed forward network One hidden layer with three nodes Back propagation learning algorithm
Napolitano et al., 1999	Fault-tolerance control	Military aircraft	1. Elevator, aileron, and rudder 2. Pitch, roll, and yaw rates	Sensor and actuator failure: - Detection - Identification - Accommodation	1. Actuator -NN to estimate angular velocity (detection) - Failure identification through cross-correlation functions - NN controllers for pitch, roll, and yaw controls 2. Sensor -A main NN (MNN) -A set of ‘n’ NNs, decentralized for ‘n’ sensors
Manry et al.,	near optimal helicopter flight load synthesis (FLS)	Helicopter	1. CG F/A load factor 2. CG lateral load factor 3. CG normal load factor 4. Pitch Attitude and rate 5. Roll attitude and rate 6. Yaw rate 7. Corrected airspeed 8. Rate of climb 9. Longitudinal cyclic stick position 10. Pedal position 11. Collective stick position 12. Lateral cyclic stick position 13. Main rotor mast torque 14. Density ratio 15. F/A acceleration, transmission 16. Lateral acceleration, transmission 17. Vertical acceleration, transmission 18. Left hand	1. Fore/aft cyclic boost tube oscillatory axial load (OAL) 2. Lateral cyclic boost tube OAL 3. Collective boost tube OAL 4. Main rotor (MR) pitch link OAL 5. MR mast oscillatory perpendicular bending st. 6. MR yoke oscillatory beam bending sta. 7. MR blade oscillatory beam bending sta. 8. MR yoke oscillatory chord bending sta. 9. Resultant mast bending sta. position	1. Modular neural network 2. Cramer-Rao Maximum a -posteriori bounds 3. Multilayer perceptron network

			forward and aft pylon links 19. Right hand forward and aft pylon links		
Yan et al., 1999	Fault tolerant flight controller	Fighter aircraft	1. Pilot pitch rate 2. Velocity commands	1. Pitch rate 2. Actuator sluggishness 3. Velocity	1. PID controller 2. MRAN controller 3. Plant
Urnes et al., 2001	Damage adaptive flight control system	Fighter aircraft	1. Mach 2. Altitude 3. Alpha 4. ABS 5. Collective Stab 6. Collective rudder 7. Differential aileron		

APPENDIX C.

References	Type of control	Application	Input	Output	Architecture
Oussedik et al., 2000	Air traffic route generator	Air traffic control	1. Airspace beacons 2. Airspace sectors	Minimum distance alternative routes	1. Coding 2. Mutation 3. Crossover 4. Sharing
Austin and Jacobs, 2001	Longitudinal flight control	Hypersonic aircraft	1. Attitude 2. Flight trajectory	Configure the control surface, along with a fixed and preset control structure	1. Initial evolutionary reproduction process 2. Floating point encoding 3. Sigma-truncation and linear scaling fitness function 4. Uniform arithmetic crossover with adaptive direction of mutation
Mulgund et al., 1998	Optimization of large-scale air combat tactics	Fighter aircrafts	1. A set of commonly-used element and division formation along with underlying tactical maneuvers and attack tactics 2. A set of principles for aggregating the small formation tactics for large engagements	1. Individual maneuverability 2. Formation tactics 3. Division tactics	1. Stochastic coding 2. Tactics implementation 3. Fitness function on the basis of friend/enemy loss, separation criteria, relative advantage, and risk assessment
Marin et al., 1999	Guidance	Unmanned aerial vehicle	Terrain data on vegetation, slope, hydrology, roads, and obstacles	Detection and monitoring of targets	SAMUEL evolutionary learning system
Nyongesa et al., 2001	Control of delay-time of anti-air missile	Ground anti-air missile vehicle	1. Missile angle 2. Missile distance	Optimization of delay time of fuze to kill the target	1. A set of eight functions 2. A set of three terminals 3. Genetic programming parameters; population, generation, mutation, migration frequency and rate 4. Fitness as success predicate

APPENDIX D.

References	Type of control	Application	Input	Output	Architecture
Zein-Sabatto and Zheng, 1997	Intelligent flight control	XCell-30 mini helicopter	1. Flight altitude 2. Rotor speed 3. Blade angle 4. Aerodynamic theory	1. Rotor speed 2. Blade angle	1. A three-layer feedforward NN: to learn helicopter dynamics 2. Genetic algorithms: to optimize coefficients for PID controller 3. PID controller: blade angle control 4. Fuzzy controller: rotor speed control
Idan et al., 2001	Fault-tolerant control	Numerical model of B 777	Aerodynamic and propulsion actuator failure information	Control of: -Speed -Pitch rate -Thrust	1. Online Neural network to learn fault tolerance 2. Pseudo-control hedging to address adaptation difficulties 3. Nonlinear single hidden layer NN to compensate for inversion error
Joshi and Valasek, 1999	Bang-bang type vortex flow control	X-29A	1. Angle of attack 2. Sideslip angle 3. Deflection 4. Bank angle 5. Heading angle	1. Close-loop system performance 2. Activity level of VFC nozzles 3. Ease of controller synthesis 4. Time required to synthesize controller	1. Model predictive variable structure 2. Fuzzy logic 3. Neural networks
Melin and Castillo, 2002	Aircraft dynamics control	General	1. Wind velocity 2. Inertia moment	1. Altitude 2. Trajectory estimation	1. Fractal module 2. Fuzzy rule base for modeling 3. NN for control 4. NN for identification 5. Aircraft dynamic system
Kewley and Embrechts (1998	Positioning military combat units for optimum performance	Military ground units	1. Organization and course of action data 2. Vehicle data 3. Map data	Estimation of enemy and friendly losses	1. A simulation model to evaluate solutions 2. A fuzzy logic module to map simulation outputs 3. A genetic algorithm to search the terrain for near-optimal combination of

					unit positions
Ganesh, 1999	Uncertainty management	Submarine	1. Platform identification 2. Normalized blade-rate	Contact speed	1. Two input single output speed fuzzy inference system 2. Target motion analysis
Wills et al., 2001	Control of complex system	X-cell helicopter		1. Outer loop: flight trajectory 2. Inner loop: pitch, roll, and yaw	1. High level control: situation awareness, reactive control, and model selection 2. Mid level control: mode transition 3. Low level control: stability and control, and augmentation system 4. Open control platform with multiplayer application programmer interfaces 5. PID controller 8. Neural network based controller
Leitner et al., 1998	Trajectory tracking control	Rotorcraft		1. Outer loop: pitch and roll 2. Inner loop: moment controls of lateral and longitudinal cyclic, and tail rotor collective pitch	1. PID controller 2. Neural network based controller